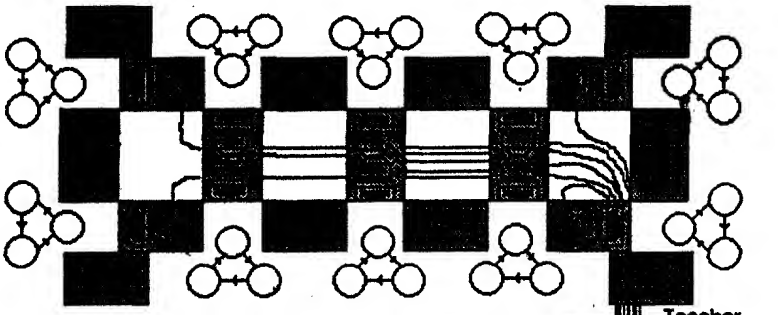
the world of

68' micros

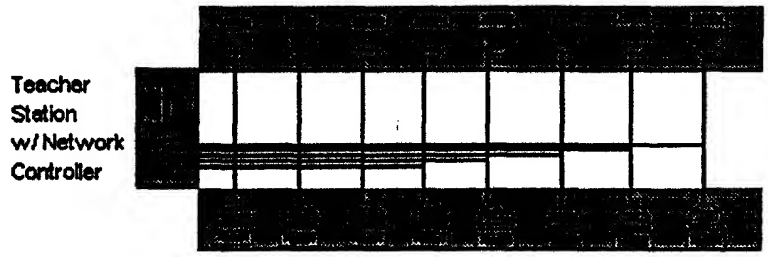
Support for Motorola Processors

15 March 1994 | *Vol. 1 Number 6* | *$4.50 Canada, $4.00 US*

The Tandy Color Computer-
an educational tool to this day....



"Pod" Arrangement - 10 Computers, 30 Students



Complete 16 Computer/Student Arrangement

CONTENTS

POSTMASTER:
If undeliverable return to:
FARNA *Systems* PB
Box 321
Warner Robins, GA 31099

the world of 68' micros

Published by:
FARNA Systems
P.O. Box 321
Warner Robins, GA 31099-0321

Editor: F. G. Swygert

Subscriptions:
$23/year (8 issues) US; $30/year for Canada/Mexico ($12 US, $16 C/M for six months- four issues). Overseas $35/year ($18 for four issues) surface. Add $8/year, $4 six months for air mail.

Advertising Rates:
Personal Classifieds- $0.20/word, $4 minimum. Name, address, and phone number free. FREE for subscribers.
Display Ads- $15 1/6 page, $20 1/4 page, $35 1/2 page, $60 full page, copy ready. Add $10 for special placement, $10 for typesetting ($5 1/4 or less). Dot matrix will be typeset if deemed unacceptable and submitter billed. 10% discount for four or more appearances.

The publisher welcomes any and all contributions. Published material becomes the property of FARNA Systems unless other, prior arrangements made. Submission constitutes warranty on part of the author that the work is original and not copywritten by another party. All opinions expressed herein are those of the individual writers, not necessarily the publisher or editor. FARNA Systems reserves the right to edit or reject any submitted material without explanation. Renumeration discussed on an individual basis.

Back issues are $4 per copy. Overseas add $1 each for airmail delivery.

Newsstand/bulk orders available. Dealers should contact the publisher for details.

Problems with delivery, change of address, subscriptions, or advertisers should be sent to the publisher with a short description.

The publisher is available for comment via Internet (dsrtfox@Delphi.com) or Delphi E-mail (DSRTFOX). The CoCo and OS-9 SIGs on Delphi are also frequented (The Delphi SIGs are sponsored by Falsoft).

The editor speaks...

F.G. Swygert

Back in December I announced a little subscription contest, where any state that brought the subscription percentage up by 2% would receive a prize. Well, I had TWO states win- Nebraska and Iowa. If you live in one of these states and subscribed or renewed in December or January, you can have your choice of (1) a free issue of "microdisk"- any over the first volume (01 Aug 93 - 01 Aug 94); (2) an issue added to your subscription; or (3) one free back issue. The back issue will be sent third class with the next mailing OR you can send $1 to cover first class postage. **In order to claim your prize, please contact "68' micros" by 15 April.**

There was one common item between the two winning states- both are the home states of Color Computer clubs. Most readers should be familiar with Mid Iowa & Country CoCo Club, they have been advertising with us since the magazine started. How many have heard of the Middle America Color Computer Club (MACCC) of Nebraska? I hadn't, not until I received an envelope with a single check and four identical forms (except for names and addresses!) for as many subscriptions! A few weeks after receiving the first such envelope, I received another, again with four subscriptions. I only received one other subscription from Nebraska. MACCC can be contacted through Bruce Gerst, 4144 U Street, Omaha, NE 68107-3141. Bruce also hangs out on Delphi.

Third class mailing seems to be working out just fine. I polled subscribers on Delphi and found that most received their magazines during the first week of February, with the latest getting in home on the eighth. The first week was the target, so all appears well. There appears to be an average 6-7 day spread across the country rather than 3-4 days with first class mail. I sent the magazines out a week earlier than with first class, and will continue to do so.

In the January issue of OS-9 Underground, Alan Sheltra commented that "one publication is going to third class mailing... I looked into the idea... and totally rejected it!... the cost difference isn't worth the headaches the cheaper way would cause. Besides, first class mags, gotta go first class!"

I thought long and hard before making the following comments. First of all, I know that Alan meant nothing against "68' micros" by the last statement. A more sensitive person might construe the remark to mean 268'm must be 2nd or 3rd class, but I know Alan better than that, and don't want anyone else to think he might have meant anything of the sort (the only reason I mention it at all).

I do contend that Alan hasn't looked into third class mail very thoroughly. There isn't a lot of hassle , and the savings are significant if you have a sizable clientele. It costs me 75 cents to send 268'm first class and about 25 cents for third class. That's 50 cents x near 300 subscriptions every six weeks, or $1200.00 per year... and the subscriber base is still growing! What it really means is I no longer have to work for free, there won't be any increase in price in the near future for subscriptions or ads, and I can ad pages with little cost increase to me.

Now the Underground isn't the same size as 268'm... it only costs Alan 52 cents per issue to mail, but he mails 12 issues per year. He would save $3.24 per subscriber per year by using third class mail, or $972 yearly based on 300 subscribers (I have no idea how many subscribers he has, just used 300 for comparison). The Underground weighs two ounces as is, meaning the only way an increase in the number of pages can occur is if a thinner paper is used or a 23 cent increase in shipping is incurred. A 32 page issue of 268'm is just under three ounces. Third class mail allows 3.3 ounces for the basic rate. I can add four pages with no additional postage- maybe six. It will cost me another three cents each if I go up to a four ounce issue (about 46 pages).

What trouble is third class mailing? Well, it costs $75 for a permit (annual fee). If you want to use an imprint rather than pre-cancelled stamps, that's another $75 (one time imprint "license" fee). The sender has to then sort all mail into bundles of no less than 10 each by state. States with less than 10 issues get put with others in a mixed state bundle. That's about it, unless there are enough to be sorted by 3/5 digit zip codes (with an even lower rate). With 300 subscriptions all over the country, that's nearly impossible. My mailing label program (DML9 by Bob van der Poel) sorts by zip anyway, so there is little extra work to do. When I deliver to the post office, all the bundles are placed in a bag and sent directly to the regional mail handling center- in my case Atlanta- where it is distributed to the states and mixed states are sorted.

There are really only two drawbacks to using third class mail. There is an extra week required for delivery (I moved my production schedule up a week to compensate) and deliveries are made over a two week period. The first customer may get his issue on the first, the last on the 15th.

But is nearly $1000 yearly and the flexibility to expand without a cost increase worth it? 268'm is as much your magazine as mine, and I promised long ago to give you as much as possible for your money...

< 268'm >

Letters to the Editor

OS-9 Level I, File Transfers to PC

I have two second-hand CoCos and OS-9 version 1.0.0 (disk but missing the manuals). I would like to obtain a 64 or 80 column text driver for the existing VDG.

I am also interested in the serial port and perhaps a hardware project to facilitate file transfer to/from TRS-80 Models 4 & 4P as well as the IBM PC/XT.

Douglas P. Beattie, Jr.
P.O. Box 47
Oak Harbor, WA 98277-0047

Can anyone help Douglas with a set of manuals or package for OS-9 Level I and the expanded display driver? I believe there was a 51 column driver available at one time (software).

But Doug, you might be better off to try finding an 80 column driver and a video monitor driver. I plan on running a hardware article for adding a monitor driver to a CoCo 1/2, but don't know about an OS-9 software driver that would supply 80 columns even then.

Your second interest is easy to answer-use a null-modem cable. *You need terminal (modem) software for each computer. Set the baud rate and other parameters the same for both (start with 1200, then increase after you get everything working properly). The cable is pretty easy. Connect pins 6, 8, & 20 on the IBM or Model 4 side to pin 1 (CD) on the CoCo. Connect CoCo pin 3 to pin 7 (grounds), CC pin 2 to pin 2 (receive to transmit), and CC pin 4 to pin 3 (transmit to receive). Now you should be able to transfer files. You can transfer ANY program, file, etc., but the only ones that each computer can read will be those in ASCII format.*

Color Computer Hard Drives

Would like to congratulate you on the great magazine you have created. Have enjoyed all issues so far from the very interesting articles to the many advertisers.

I might mention a subject that has had very little coverage in the past years and that is the Hard Drive (for the Color Computer). Such as the dos and don'ts, setting up directories, using DECB and OS-9 on the same drive, why some programs won't work from a hard drive, etc. I have two hard drives but wish I had more info.

I've enclosed $8 to keep my subscription coming first class- third class takes about a month to get here!

David Garner
1201 Wilder Ave., #1503
Honolulu, Hawaii 96822

David, I am planning a hard drive issue some time this year- just as soon as I get all my information gathered! How about sending me a letter describing your system, how you have it set up, and the problems you encountered setting up the hard drives. That itself would make an interesting article, and perhaps I can find answers to some of the problems you had and are having now to supplement your writing.

I'm keeping Alaska, Hawaii. APO, and FPO addresses on first class for now at no charge due to the additional time it takes for even first class mail to get there. Until the number of those subscribers reaches the point where it is an added burden on finances, I won't be charging extra- am returning your check. You should be getting your magazine just a few days before the third class issues in the 48 states

New England/Canada 'Fest?

Thank you for supporting the CoCo Community like you do. Don't you think it is about time for us to let Tandy know that the CoCo is still alive, even after it's "official" death five years ago? I'm happy to see someone is going to continue the good work of "the Rainbow".

I went to one Rainbowfest in New Jersey and really liked it. Do you think we could see a CoCoFest in this region in the future? It's a bit closer to Montreal than Chicago or Atlanta!

Serge Alepins
141 Charron, Lemoyne
P.Q., Quebec I4R 2K6
CANADA

Serge, Maybe I'll send the Tandy support division a free copy. That way they could at least let people know where to get additional support and software.

As for a New England/Quebec area CoCoFest, well, that is really up to the area clubs and dealers. I will help coordinate- if anyone in the New England region is interested in putting on a CoCo/ OS-9/OSK show...

Spare CoCo3 - Transfer to IBM

I saw your ad in "Nuts & Volts". I have a CoCo 3 that I no longer use (2 drives) since I bought an IBM platform. I also have Rainbow issues from 84 to 90, and a green monitor. If any of your readers need a spare CoCo, they can get in touch with me.

I need a source for the program "Xenocopy" or some other program that will allow me to transfer text files from my CoCo disks to the IBM, preferably something that runs on the IBM. Microcom used to sell Xenocopy, but are now out of business.

George W. Sturm
HC 62 Box 183K
Durant, OK 74701-9131

Anyone know a source for Xenocopy or maybe Elite Transfer? If so let me know, and I'll pass it along to George, or let him know yourself. No prices on the CoCo were given.

Goof on MD #2!

I have a problem with the contents of vol. 1 # 2 microdisk. When I attempt to use TC3 to decompress GETERM it goes to work for a while then stopped with a "File to Large" error. On page 5 of issue #2 you say there is a third version of TC, called TC31, which handles larger files. Do I need this version to decompress GETERM?

I look forward to each new issue of 268'm , and so far have learned a lot from each one too. Thanks for publishing it.

Richard Bair
335 Jefferson Ave.
Glencoe, IL 60022

Richard, I goodfed! GETERM was indeed compressed with TC31. You will find TC31 and a new archive of GETERM on this issue's microdisk. The new archive is made with TC3, not TC31. TC31 requires a 512K CoCo 3, TC3 runs fine with a 128K model. The reason you got a "file to large" error is that TC uses all the memory for buffer space. Since TC31 uses more buffer space, the "buffer block" was to large (both TC versions have fixed buffer sizes). I also corrected the mistake on MD #2 for future purchasers.

Lookin' Good

Frank, each issue looks better than the last. Your lay-out skills are getting better with each issue. In the over-all view, "268'm" is coming along just fine- good job!

James DeStafeno
RR1 Box 375
Camden, Wyoming, DE 19934

James is the former publisher of "The 68xxx Machines". I thank you for the good comments James!

< 268'm >

Letters are printed on a space available and popular subject matter basis. If you don't want your letter printed, or wish to withhold your address and/or name, please state so when writing. In some cases, letters are edited for space and/or clarity. If a personal reply is desired, please enclose an SASE.

The Educational Color Computer

F.G. Swygert

The CoCo2 in the classroom- it could happen again with your help!

First, some background...

Tandy started their classroom infusion with the Model I in 1977. The first software package available was "Math-1", a math facts primer developed by a Fort Worth teacher for grammar school students (basics through multiplication and division). At this point Tandy had yet to have any real competition in the educational markey- the Apple wasn't even developed! Onthe down side, however, was the cost and maintenance of several Model I systems (anyone who has used a Model I knows that it wasn't the easiest to use nor most reliable system, but it was basically the ONLY complete system available at a price larger schools could afford) and the fact that not many educators were sure the computer had a place in the classroom. Other software soon followed, much of it centered on using computers. Tandy did create some inroads into classrooms, however, and developed special educational software and components for their computers. Then the Apple II came along and usurped Tandy's educational monopoly.

The Apple II computer made such an impact on the home computer and educational markets for two reasons (has anyone been to a school that didn't have Apple IIs at one time or another?). The Apple II was the first practical, inexpensive, easy to use, all-in-one computer available. The second reason is that Steve Jobs and Mike Wozniak, both college students when the Apple II was developed by them, reasoned that if they could get schools to buy Apples, then the students and teachers would want similar computers for themselves. So Apple sold schools Apple II systems at very deep discounts, often at cost. The result was just as they predicted- the Apple II became popular among studentsof all ages across the country.

With the arrival of the Color Computer in 1980, Tandy had a real competitor for the low cost Apple II. In many ways, the CoCo was better than the AppleII- it had a more modern microprocessor and was better priced. The only advantage of the Apple was its internal expansion slots, but then the CoCo already had all necessary ports (for the time!) built right in.

One Disk Drive, Many Computers!

The first hardware item developed by Tandy specifically for the educational market was the Network Controller I (NCI- catalog #26-1210). This was a simple switchboxthat allowed connecting 17 Model I computers together- 16 student stations and one central station. Until this time, a teacher had to physically carry a tape (or, rarely, disk- a floppy drive was $499 and the required expansion interface $299, alomost $800 per computer!) to each students' computer and load it if some technique was to be demonstrated or the same program used by each student.

The 17 computers were connected through the cassette cables to the NCI. The central station was the only one required to have a disk controller. Any program loaded into the central station could be transferred to any or all student stations. The teacher simply asked all students to receive a program to type "CLOAD" on their stations, then typed "CSAVE" on the central station. For a student to save a program to disk, the input select rotary switch was turned to the station and the opposite repeated (only one station could save at once!).

The NCI required a good deal of coordination by the teacher, but was faster than carrying a single cassette around to each computer. In a 1982 "TRS-80 Microcomputer News" article describing use of an NCI, a teacher comments that it was helpful not to have to keep up with the whereabouts of 16 cassette recorders and tapes all the time.

The NCI was updated in 1981 with the introduction of the Network Controller II (NCII- catalog #26-1211). This was basically the same thing with updates to allow use of the higher tape speed of the Model III and Color Computer (1500 baud compared to the Model Is 500 baud) and the use of a cassette recorder for the central station. A switch was added to the unit to allow switching the central unit from cassette to disk, so both could be used. The cassette recorder plugged into the NCII- the central station connected to the NCII as before.

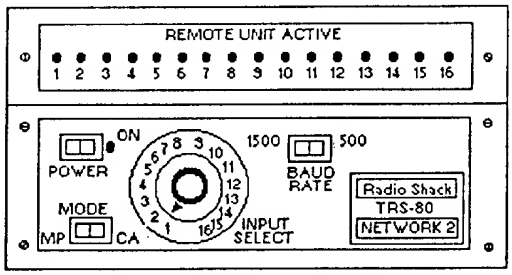
The NCII could be used with the Model I/III/4, Color Computer. or even the Model 100. Model I/III/4 computers could be mixed. When using the CoCo or Model 100, the central and student stations had to be the same. If only ASCII files were to be transferred, a Model I/III/4 or CoCo could be used with Model 100s. The NCII was discontinued after 1987.

The Network Controller III, made in 1982, was the last in the series. NCIII was only for the Model III/4, and operated more like a directly connected BBS system. Cassette ports were still used.

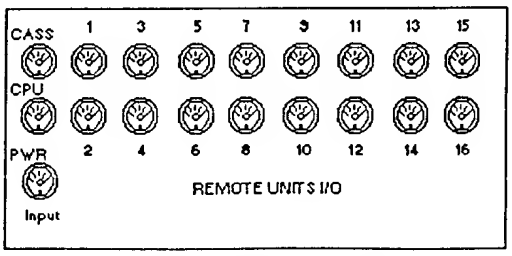
A Special CoCo 2

When the CoCo 2 was introduced in 1983, a special educational model was produced. This was basically a standard CoCo 2 with one exception- there was no TV RF output. Instead of an RF modulator, these units contained a video amplifier for driving a composite monitor. This change was made mainly to create a clearer screen. Many of you remember (or know now!) that the RF signal isn't the best shielded. A better cable helps (cable TV type coax is best), butthere is still some screen interference. One can easily imagine how much interference could be generated by 17 CoCos closely clustered together! The composite output solved this problem easily. A minor drawback to the monitor driver was that there was no provision for sound output- a minor consideration for an educational system. The Tandy monitor driver used a 1/8" headphone type jack rather than the more common RCA type connector, which was used by the monitor itself.

A Tandy service document I have (technical bulletin CC:27, dated July 20, 1984) illustrates the installation of the "Direct Video II" composite monitor driver. I'm not



Network Controller 2 control panel.



Network Controller 2 rear panel.

sure if this indicates that a composite driver was made for the original CoCo, that this was just an updated version of the original CoCo 2 monitor driver, or it was designated with a "II" simply because it was only for the CoCo 2 (referred to as "CoCo II" in the service bulletin). If the later is tha case, this would indicate that the monitor driver was introduced in mid 1984. If anyone has seen a Tandy installed monitor driver (in place of the RF box) in an original CoCo or TDP-100, please let us know!

The CoCo in Atlanta

Some of the Atalanta, Georgia, school districts acquired Tandy Color Computer 2s and Network Controller IIs in the early 80s. All of these had composite monochrome monitors and 64K. In the later part of the decade, 1989-90, the schools started updating there computer labs with IBM compatible systems. This created a surplus of the CoCos, which were sold at auction. The Atlanta Computer Society (ACS- a large CoCo club, sponsors Atlanta CoCoFest) knew of the auction, and several members wnet, but the systems were lumped into large lots- to many for anyone in the group to purchase.

One particular member, who desires to be unnamed, assisted the schools in setting up their new systems. In return, he requested one palette of CoCo 2 systems. Those he started giving to new ACS members.

Well, this fellow also happened to know the person who ended up with the entire lot of Color Computer equipment from the school auction. As our CoCo benefactors original lot of systems ran down, he purchased more from the person who bough t the lot.

Eventually, our benefactor ended up with the majority of the school systems, NCIIs, CoCos, monitors, and all. It seems nearly everyone in Atlanta who wants a CoCo 2 now has one (or more!), and there are about 200 left!

I first saw the person with these machines at the Atlanta CoCoFest in 1992. He had rented a booth where he was GIVING AWAY CoCo 2s. This was a surprise to many of us dealers, some of whom had CoCo 2s they were trying to sell (luckily, I didn't have any!).

I met him again in 1993, doing the same thing. This time, however, he had one of the Network Controller IIs on display. This time I found the time to approach and talk with him about the history of the computers, and learned how many he had. Since I had just started this magazine, it seemed I might be able to help him find a good home for the remaining machines.

Is there a school out there that could use these? They are (almost) free! Well, the computers and monitors ARE free, but shipping will have to be paid. I estimate it will cost about $12 each (monitor and computer) to pack and send UPS Ground anywhere in the 48 states .

If anyone is interested, some Network Controller II units and cables are also available. These are the only things that our benefactor has any money tied up in, and he would like $50 each for the controllers and cables. A complete set of 17 CoCo 2s, monitors, Network II controller, and cables can be purchased for only $254. I'll also include copies of several "TRS-80 Microcomputer News" articles I have describing different NCI & II setups and uses. If you want copies of the documentation and articles before making a decision, send $5 to cover costs.

The only other item possibly needed (but not absolutely necessary) would be a disk drive and controller for the "master" CoCo. A $300 computer lab for teaching computer basics and the BASIC programming language. This would be a good club or community project- raise money and set up a computer lab for some small school, and help support it.

For my part, I 'll help check the computers before sending and do all the packing and shipping- free of charge. Our benefactor is retired and can't pack, ship, and distribute these systems- so I volunteered.

The computers are all well used and show a little age with discolored cases, but all are in reasonably good shape. I have to make a two hour journey (one way) to get these things, so please be patient when requesting them. There are no guarantees, but an extra CoCo or two will be sent along. All systems will be tested and verified to work before shipping.

Because of the efforts required to pick up and send, please make requests for at least five systems and monitors. There aren't enough monitors for quite all systems, so we can ship computers without monitors (but not the other way around).

Help your community- find a home for these computers and networks! They would make a great community education project, especially for underpriveleged areas and students, or schools with minimal funds.

< 268'm >

The HD6309E CPU

Robert Gault

What is it and why should I care? PART 1

Most Coco owners know that somewhere in the guts of their favorite computor is a plastic bug with forty legs called a Central Processing Unit (CPU). You might even know that Motorola is the manufacturer. This device has the Motorola part number MC6809E (MC68B09E in the CoCo 3) and is described as a "high-density N-channel, silicon-gate". It is an HMOS (high-density metallic oxide semiconductor) device.

For some reason, Motorola decided that a low power version of the 6809 was required and licensed Hitachi to produce a "Complementary MOS" (CMOS) unit. The result was the HD6309 (the HD stands for Hitachi). The CoCo uses an external clock version, the HD6309E, and the CoCo 3 a 2MHz version, the HD63B09E. Throughout this article I will refer to all version as the 6309.

If you read the Hitachi specs for this unit (Hitachi 8/16-Bit Microprocessor Data Book), the only difference you could find between the 6809 and 6309 would be that the power consumption and thus heat output is lower in the Hitachi unit.

Two significant problems with all Cocos have been excessive heat generation and marginal power supplies. A Coco3 with 512K of added memory may well require a fan installed to prevent overheating. Any change that can reduce the Coco power load and heat output is therefore welcome; the 6309 fits the bill perfectly.

Should you and can you make the change in your Coco? If you have overheating problems, making the change will help but a installation of a fan would be better. If you don't have problems, everything else being equal, I would say don't bother.

The CPU is not socketed in the CoCo 2 and 3 but is soldered to the printed circuit board. Removal of the 6809 and installation of a socket for the new 6309 can be fraught with danger for the beginning solderer. There is a very good chance of destroying the trace lines on the board and in consequence the computer.

Happily there are other reasons for wanting a 6309 besides low power consumption. I have been told by professionals in the computer industry that there is a little publicized but common practice when creating revisions of integrated circuits like the 6309. The designers make every effort to shrink the size of the circuit. The smaller the circuit, the less it costs to produce in quantity.

However, the designers can't resist the challenge of filling up the newly available space with enhancements which in this case were not made public since they were not part of the licensing aggreements between Hitachi and Motorola. This is exactly what happened with the 6309.

If you read the last few issues of "Rainbow" or have been following the technical message bases on a big BBS like Delphi, you will have seen that Japanese computer hobbiests discovered many new functions in the 6309 plus a new mode of operation in which all instruction cycle counts have been decreased. I won't rehash this information but leave it to the reader to pursue.

Should this additional information change your decision to swap CPUs? I am afraid the answer for most Cocoists is still a firm maybe. There is no software advantage for users of the standard Basic ROMs or standard OS-9. There is some advantage if you purchase modifications to OS-9 from Burke&Burke (Power Boost) or Gale Force Enterprises (NitrOS-9). This advantage will be in system overhead but not directly in any application programs. Users have, however, reported a noticeable increase in speed (20% on average), so if you use OS-9 a lot consider one of these enhancements.

The real advantage of the 6309 will be realized only by assembly language programmers. This is because there is just not enough software available which utilizes the 6309 and there is not likely to be. As much as I like the CoCo, it does not take much looking to see that customer support has pretty much evaporated.

In the hopes that I might entice a few readers to become assembly language programmers, I shall present a few articles which make use of the new 6309 opcodes and compare speed advantages, if any, over the 6809. While it is possible to write assembly code with existing assemblers, it is much easier to use an editor/assembler which understands the new 6309 code. Whether we program under Disk Basic or OS-9 makes little difference, but I beleive that the ideal platform for experimenting with the new CPU is Disk Basic; a single user - single tasking - environment.

The only Disk Basic editor/assembler I know of that is 6309 ready is EDTASM6309, my own patch package for Tandy's Disk EDTASM+. I will unabashedly use the EDTASM6309 format for most of my examples. Readers may inquire about the availability of EDTASM6309 (you must already own Disk EDTASM+) by writing to me.

To follow this series of articles, you will need a copy of "A Memo On The Secret Features of the 6309" by Hirotsugu Kakugawa or some other paper listing the features of the 6309. If you have purchased the complete packages from Burke&Burke or Gale Force Enterprises, or EDTASM6309 from me, you should have this information (see sidebar)

Let's start with a short routine to see which CPU is presently in operation. Next time we'll start looking at the Coco3 ROMs and ways of speeding up operation with the new 6309 code.

You will need a method of testing which CPU is present because unexpected and disasterous results could occure if a 6809 tries to run 6309 code. This is a good place to explain one of the significant differences between the two CPUs. When the 6809 sees an unknown instruction (opcode), it skips over the offending code, doing nothing until a good opcode can be found. The 6309 does something very different. A bad/unknown opcode causes the 6309 to execute an error trap routine. The error routine pushes all registers onto the reg.S system stack, sets a new register (MD) with the error type, and jumps to the address stored at $FFF0. Woe unto the unlucky programmer who tries to execute an illegal code without setting up the $FFF0 error vector.

The code below attempts to execute an instruction legal for the 6309 but illegal for the 6809. Note how a 6309 CPU would see the instruction CLRD (with the result that register A and B both clear) while a 6809 CPU would see an unexecutable code and skip to the next instruction, CLRA and only clear register A. Both listings assemble identically.

Summary of HD6309 Features

More registers:

Two 8bit accumulators.
One 16bit concatenated register
One 16bit value register.
One 8bit mode/error register.
One 32bit concatenated register

Twomodes:

MC68B09E emulation mode and HD63B09EP native mode. Reduced execution cycles when running in native mode.

Additional Registers

The 6309 has 6 additional registers. Only 4 of those are actual registers. The other 2 are combinations of registers. Those registers are:

ACCE - 8 bit accumulator.
ACCF - 8 bit accumulator.
W - 16 bit concatenated register (ACCE and ACCF combined).
V - 16 bit register (which can only be accessed with the inter-register instructions).

continued on next page

Listing 1:

```
00100 * This routine tests for the presence of a
6309 CPU and indicates
00110 * whether it is present. The routine is
written to EXECute under
00120 * Disk Basic. This is the 6309 version.
00130
00140     title 6309CPU Test
00150
00160 printS equ   $b99c      send string
indexed by reg.X to consol
00170 cr    equ   $0d         carriage return
00180
00190     org $7000           upper RAM
00200
00210 start  ldb #$ff         initialize reg.B to a
non-zero value
00220 * this next instruction is a 6309 opcode
00230     clrd                clear reg.D
00240     leax  mes63-1,pcr   point to 6309
present message
00250     tstb                does reg.B=0 ?
00260     beq   a@            yes, print message
to screen
00270     leax  mes68-1,pcr   point to 6809
preent message
00280 a@  jmp   printS        print message and
return to Basic
00290
00300 mes63  fcb   cr
00310     fcc   /There is an HD6309 present./
00320     fcb   cr,0          * trailing 0 = end of
message
00330
00340 mes68  fcb   cr
00350     fcc   /There is an MC6809 present./
00360     fcb   cr,0
00370
00380     end   start
```

Listing 2:

```
00100 * This routine tests for the presence of a
6309 CPU and indicates
00110 * whether it is present. The routine is
written to EXECute under
00120 * Disk Basic. This is the 6809 version.
00130
00140     title 6309CPU Test
00150
00160 printS equ   $b99c      send string
indexed by reg.X to consol
00170 cr    equ   $0d         carriage return
00180
00190     org $1D1            cassette buffer area
00200
00210 start  ldb #$ff         initialize reg.B to a
non-zero value
00220 * The next two lines form an 6309 opcode
but the 6809 only sees clra.
00230     fcb   $10           skipped by 6809
00240     clra                clear reg.A
00250     leax  mes63-1,pcr   point to 6309
present message
00260     tstb                does reg.B=0 ?
00270     beq   a@            yes, print message
to screen
00280     leax  mes68-1,pcr   point to 6809
preent message
00290 a@  jmp   printS        print message and
return to Basic
00300
00310 mes63  fcb   cr
00320     fcc   /There is an HD6309 present./
00330     fcb   cr,0          * trailing 0 = end of
message
00340
00350 mes68  fcb   cr
00360     fcc   /There is an MC6809 present./
00370     fcb   cr,0
00380
00390     end   start
```

If you don't have an editor/assembler, the following Basic program will poke the required code into memory.

```
10 REM CPUTEST. INCLUDE LINE 60
IF DESIRED. USE BY "EXEC&H7000"
20 LI=80
30 FOR M=&H7000 TO &H704D
STEP10:SUM=0
40 FOR I=0TO9:READ A$:VA=VAL
("&H"+A$): SUM=SUM+ VA: POKE
M+I,VA:NEXT:READ CHK: IF SUM <>
CHK THEN PRINT"ERROR IN LINE"
LI:END
50 LI=LI+10:NEXT
60 REM SAVEM "CPUTEST",&H7000,
&H704D, &H7000
70 END
80 DATA C6, FF, 10, 4F, 30, 8D, 0 , 9 ,5D,
27, 878
90 DATA 4 , 30, 8D, 0 , 20, 7E, B9, 9C, D ,
54, 789
100 DATA 68, 65, 72, 65, 20, 69, 73, 20, 61,
6E, 911
110 DATA 20, 48, 44, 36, 33, 30, 39, 20, 70,
72, 640
120 DATA 65, 73, 65, 6E, 74, 2E, D , 0 , D
, 54, 699
130 DATA 68, 65, 72, 65, 20, 69, 73, 20, 61,
6E, 911
140 DATA 20, 4D, 43, 36, 38, 30, 39, 20,70,
72, 649
150 DATA 65, 73, 65, 6E, 74, 2E, D , 0 ,00,
00, 602
```

< 268'm >

Additional Registers (cont'd)

MD - 8 bit mode/error register.

Q - 32 bit concatenated register (ACCA, ACCB ,ACCE and ACCF combined).

Additional Instructions:

Error trapping of illegal instructions and zero divisions. The new instruction set can be used in both native and emulation mode.

ADCD - Adds immediate or memory operand to the D register plus the current status of the carry with the result going to D.

ADCR - Adds two registers together plus the current status of the carry.

ADDE , ADDF, ADDW - Add of immediete or memory operand to E, F or W with results going to E, F or W.

ADDR - Adds two registers together.

ANDD - Logical AND of immediate or memory operand to D register with result going to D.

ANDR - Logical AND of a register with the contents of another register.

ASLD - Arithmetic shift left of the D register filling the LSB.

ASRD - Arithmetic shift right with sign extending.

BITD - Test any bit or bits of the D register.

BITMD - Test any bit or bits of the MD (mode) register.

CLRD, CLRE, CLRF, CLRW - Clear register D, E, F or W to zero.

CMPE, CMPF, CMPW - Compares the contents of E, F or W with the immediate or memory operand. Sets all CC except H on result.

CMPR - Compares one register to another and sets all CC bits except H on result.

COMD, COME, COMF, COMW - One's compliments D ,E, F, or W. Changes all zero's to one's and all one's to zero's.

DECD, DECE, DECF, DECW - Decrement 1 from D, E, F, or W.

DIVD, DIVQ - Does a 16 bit by 8 bit (DIVD) or a 32 bit by 16 bit (DIVQ) signed divide with immediate or memory operand with quotient in W and modulo (remainder) in D.

EORD - Logical exclusive OR of D and immediate or memory operand.

EORR - Logical exclusive OR of one register with the value of another register.

INCD, INCE, INCF, INCW - Increment D, E, F or W by 1.

LDE, LDF, LDQ, LDW, LDMD - Standard loading of E, F, Q, W or MD with immediate data value or operand from memory (LDMD only valid with direct mode).

LSLD, LSLW - Same as ASL

LSRD, LSRW - Logical shift right. Shifts D or W one bit right filling MSB with zero.

MULD - Performs as 16bit by 16bit signed multiply with immediate or operand from memory. Result stored in Q.

NEGD - Two's compliment D register.

ORD - Logical OR of register D.

ORR - Logical OR of one register with another.

PSHSW, PSHUW - Stores contents of the W register on the stack (system or user stack).

PULSW, PULUW - Pull value from stack into register W (system or user stack).

ROLD, ROLW - Rotate D or W one bit left through the Carry Condition code.

RORD, RORW - Rotate D or W one bit right through the Carry Condition code.

SBCD - Subtract an immediate or memory operand plus any borrow in Carry from contents of D. Result stored in D.

SBCR - Subtrace the value of one register from another plus and borrow in the CC carry.

SEXW - sign exdend the W register into the D register.

STE, STF, STQ, STW - Store register E, F, Q or W to memory location or two memory locations.

SUBE, SUBF, SUBW - Subtract immediate or memory operand from E, F or W. Result stored back in same register.

SUBR - Subtract the value of one register from another.

TFR (Block transfer) - Transfer W number of bytes from one location to another. Returns pointer registers offset of the starting value in the W register and returns the W register as 0. Indexed operation only.

TSTD, TSTE, TSTF, TSTW - Test contest of D, E, F or W by setting N and X condition codes based on data in register.

An edited, printed copy of "The 6309 Manual" by Chet Simpson is available from FARNA Systems. Original archive is on "microdisk".

SPOTLIGHT ON MICROWARE:

MicroWare in the Wall Street Journal

F.G. Swygert , E. Gresick, & F. Hogg

Is Microware finally going places? The following is a brief synopsis of an article from The Wall Street Journal, Tuesday, January 18, 1994, page B1. The title is "Little Microware Aims to Be a Multimedia Giant". The WSJ article was written by John J. Keller, a staff reporter. Any comments within the article enclosed by parentheses are my own.

Microware just may get a hold on the interactive TV business, expected to be in millions of homes by the turn of the century. Bell Atlantic Corp. chose Microware's OS-9 for use in it's interactive TV control boxes. The key reason for choosing OS-9 is it's ability to handle multiple real-time tasks simultaneously. Multi-tasking is seen as critical for interactive TV, letting shoppers order instantly or game players respond to TV game shows, polls, etc.

Microware was seen as a surprise contender in the highly competitive multimedia field, in which Microsoft has invested around $100 million a year for the past few years. In fact, OS-9 beat out Microsoft's product, Modular Windows, for the Bell Atlantic contract. Microsoft does not dispute this claim, but points out that it is interested in supplying a graphical user interface as well as real-time software. Bill Gates contends that early interactive TV will be disappointing because the user interface isn't good enough. Microsoft also claims that they are behind schedule because they are trying to deliver software for several segments of the market, including the set-top boxes and other interactive TV related software.

Last year Microsoft allied itself with Intel Corp. and cable box maker General Instrument Corp. in an effort to dominate the set-top market. But the alliance is behind schedule and has yet to announce a final product. There has also been resistance to Microsoft's plan to charge a fee every time someone conducts a transaction with a set-top box using Microsft software. In contrast, Microware will charge no fee over the normal licensing payment.

Microware was founded in 1977 by Kenneth Kaplan and two friends from Drake University in Des Moines, Iowa. Their initial project was an educational robot for teaching handicapped children- a project which failed.

In 1980, Motorola Inc. approached Microware about designing an operating system for a new microprocessor (the 6809... It is possible that Microware's efforts to control their robot in real time was responsible for Motorola's interest). Mr. Kaplan stated that about 80% of Microware's business today is Motorola based.

In 1982, Tandy Corp. chose OS-9 for its TRS-80 Color Computer, an early entry in home computers (This statement is rather ironic- it indicates that OS-9 was the primary operating system for the CoCo, not a secondary system that wasn't real popular with most CoCo users, only an "elite" cadre). Soon after, Fujitsu Ltd. of Japan used OS-9 in some of its computers (a dual 6309 system was shown by the Japanese delegation at the Chicago CoCoFest in 1993). In 1986, Phillips and Sony chose OS-9 for their Compact Disk-Interactive systems, largely due to Microware's proven success in large indutrial and scientific projects.

Something else also happened in 1986- Microsoft made a bid to buy Microware in March of that year. According to Microware, Microsoft saw that Sony and Phillips had chosen OS-9 for the CD-I units and Microsoft wanted a part of that market. Rather than design a new operating system, Microsoft made an offer to buy Microware. Instead, Ken Kaplan made a counter offer for a joint venture, and talks fell apart. Microsoft confirmed this account.

Microware is a closely held company and does not give out business statistics. All Mr. Kaplan would say is that Microware is about 1/100th the size of Microsoft. This indicates an approximate revenue of around $25 million annually.

Today, half of Microware's sales and 200 employees are overseas. Brian Smith, vice president of market development at Phillip's Digital Video Communications Systems Division says;"They have a nice, tight system that operates well. It's powerful enough to do real-time video functions and you can still load applications software on top." (All you OS-9 users have been saying that all along, that's why you fell in love with OS-9 in the first place, right?) OS-9 is used by the California Department of Transportation to run traffic lights, toll booths, and weigh stations; by Chrysler Corp. to design and build braking systems; and at the University of Cern in Switzerland to control the magnets in a particle accelerator (atom smasher).

This additional information was contributed by Ed Gresick, President of Delmar Co.:

Following is a brief summary of a recent bidder's meeting at Bell Atlantic about 'Video Dial Tone Service' - their name for AT&T's VOD (video over data [lines]).

Three manufacturers have been selected to provide TV set-top boxes - Philips, IBM and DiviCom. OS-9 is mandatory. Graphics and sound should be done under RAVE. All box vendors must use Motorola's 68000 series microprocessors or a compatible chip. IBM will use the 68000 with plans to upgrade to their PowerPC 400 series as soon as they are available (Motorola is developing its own separate PowerPC 300 series for this market). Philips will also be using the 68000 with an upgrade planned - no chip disclosed. DiviCom is using a 68020 microprocessor - again no info on future upgrades.

IBM will be using graphics and MPEG-1 decompression chips from outside suppliers, switching later to their own MPEG-2 and graphics chips. Philips will be using an MPEG decompression chip from Compression Labs and graphics chips based on CD-I. No info on DiviCom.

The initial test market will be Northern Virginia and involve 260,000 homes. Time frame is to start service in the summer/ fall of 1994. A second test market is planned for northern NJ - no time schedule.

The Virginia test will use parallel-processing computers from nCube. The network management and control software is from Oracle. Another future candidate is a server built around DEC's Alpha family of RISC microprocessors.

The Asynchronouse Digital Subscriber Line Data-Over Voice modem is being tested using equipment from Northern Telecom, AT&T and Westfall. No vendor had been selected at that time.

Comentary by Frank Hogg, President of Frank Hogg Laboratory, Inc.:

The deal as reported in the Wall Street Journal about Microware's fantastic deal

with Bell Atlantic is important to all OS-9 users. However it may not be directly felt. For instance, it does not seem to be like the CoCo in that people who have these 'set top boxes' may not even know they are computers much less have OS-9 as their operating systems. So what makes this deal important?

1) Microware can use this deal when negotiating deals with other companies which will help sell OS-9 to new companies thus adding to the user base. The bigger the user base, the more attractive OS-9 is to software developers and the greater the likelihood for new and exciting software we all want and need.

2) It will give OS-9 respect. In the past and even in the WSJ article the CoCo OS-9 was mentioned as a milestone for Microware. While the CoCo was certainly good for Microware it did not and does not impress the Mac and IBM using companies. This deal can be held up to that audience and be much more impressive. This deal is a milestone like the CoCo and CD-I. The fact that Microware beat out Microsoft will also be very useful in making OS-9 respectful.

3) In the years to come you can point to that 'box' on your friends TV set and say that it is run by OS-9, the same thing you use in your computer at home. The same one this friend likely criticized as not being 'main stream'.

4) I have no knowledge about the construction of the 'set top box' but it seems unlikely that it will be used as a OS-9 computer by its users. In that regard it does not seem likely to directly affect us. However, if a company was to make a home computer using OS-9 they would have an easier sell. It may even convince Phillips to come out with a CD-I player that can be used as a computer.

5) And finally, this deal, if carried thru to the 10 million sales that the WSJ article eludes too, will make Microware some serious money. That will be reflected in more and better products from them and that will eventually help all of us.

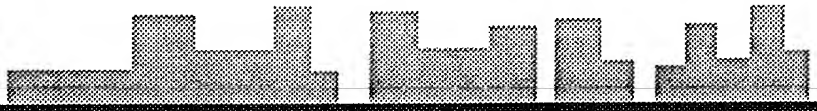
My hat is off to all of those at Microware that put this deal together. Job well done!

< 268'm >

The Glenside Color Computer Club of Illinois Presents...

The Third Annual "Last" CoCoFest!

May 21st and 22nd, 1994



at the

Holiday Inn, Elgin

A Holidome Indoor Recreation Center

345 W. River Road

RESERVATIONS: (708) 695-5000 (ask for CoCoFest rate)

RATES: $52.00 (+10% tax) per night

ADMISSION: $15.00 at door, $10.00 + SASE in advance

(or $10.00 + $0.50 postage and handling)

Send advance payments to:

George Schneeweis, Treasurer

Glenside Color Computer Club

RR #2, Box 67

Forrest, IL 61741-9629

For more info contact Tony Podraza (708-428-3576), Carl Boll (312-735-6087; CBJ@ delphi.com), or Brian Schubring (708-529-3539; theschu@delphi.com)

How to get there: Elgin is about 30 miles west of Chicago. Interstqates 88,55,57, and 94 all connect to I-294. Take 294 north to I-90, then 90 west to hwy 31. Go south on 31, then take the frontage road to the Holiday Inn (east of 31). Those using I-80 & 90 east of Chicago should get on I-94 in Indiana (at I-80/90/94 intersection) and go to I-294 north. North of Chicago, take I-94 or 90 to I-294 south, then to I-90 west.

Mid Iowa & Country CoCo Club

c/o Terry Simons

1328 48th

Des Moines, IA 50311

Do you want support tomorrow?

Stay with a group that keeps you in touch!

Annual Membership:

$16 US, $21 Canada

The "Upgrade" disk based newsletter requires a CoCo 3 with 80 column capable monitor (RGB preferred for best viewing of graphics) and at least one 40 track capable (FD-501 or third party drive) disk drive. Terry Simons is the treasurer and newsletter editor for the club. His phone number is 515-279-2576 (after 8 PM CST). Newsletter published monthly from approx. September - May, once during summer (9-10 issues per year).

The Hardware Hacker

Dr. Marty Goodman

Using a Color Computer as a Controller

This is a tale of an actually project I did recently, which nicely illustrates the use of a Color Computer as a "controller board". Recently I had occasion to be called upon to design and build a prototype of a child's toy. This toy required that three rather bright (6 volt, .5 amp flashlight type) lamps be flashed in specific fashions either periodically or whenever a button was pressed. The fashion in which the lamps flashed was to depend on the time of day, as displayed on a setable, battery-backed-up clock. Two other lamps needed to be controlled as well.

It was clear that a "controller board" was in order for this project. Because Color Computer 2's are still available to me via friends and flea markets for under $15 each, and because I have what amounts to a complete development system for such a "controller board" in the form of my own CoCo 3 system, assorted I/O cards for that system, word processors, and the Macro 80C assembler from Microworks, I decided to use the CoCo 2 as the brains of this prototype. Note that common 8751 and 68HC11 - based controller boards available commercially cost $100 or more, with the development system for same costing $500 to $1500 depending on how elaborate you want to get. Of course, a CoCo would be unacceptable for a PRODUCTION device, given its size, power consumption, and the fact that they are no longer available in large quantity because they are no longer in production. But, with my "development system" and years of experience with the software and hardware in question, the CoCo 2 made a *perfect* inexpensive controller board for a one of a kind prototype that was to demonstrate the concept for the toy, and which was not required in any way to serve as an exact design or prototype for mass production.

I required 16 lines of PIA-style ttl level I/O. I *could* have used the 15 lines available from the keyboard PIA, then scavenged an extra control line from somewhere else, for my project would not be using a keyboard. However, I decided to add an extra PIA to the CoCo. For my development system, this was trivial: I used an old Dennis Kitz parallel port card, that was really nothing other than a single PIA chip addressed to $FF50-$FF53 with all 16 general purpose lines available on its 20 pin connector.

As it happens, the B port of a 6821 PIA can happily drive directly the base of a Darlington transistor (such as the TIP 120 that Radio Shack sells). This allowed me to have the PIA controlling my 3 watt 6 volt flashlight bulbs using no more than a board with three Darlington transistors on it. I just hooked the appropriate PIA I/O lines directly to the base of three TIP 120 Darlingtons and hooked the emitter of the Darlington to ground. I then hooked one side of the flashlight bulbs to a 6.5 volt power supply (to take into account the small voltage drop across the transistor junction) and the other side of each bulb to the collector of the Darlington. The 6821 would turn the bulb on when the proper I/O line was made high, and off when that line was made low.

Another nice thing about this arrangement was that at power up or after a hardware reset, all the bulbs would be turned OFF. This Darlington was rated to handle up to 60 watts of power (you'd need a heat sink if you wanted that kind of power handling), and very similar Darlingtons (such as the TIP 121) are available to handle even more power than that. Thus, you can use this sort of arrangement for controlling multiple lines of respectable power level applications.

To implement the clock, I did something rather nutty: I bought a $10.00 Radio Shack LED, batter backed alarm clock, then took it apart and reverse-engineered the circuitry for the LED display.

The LED display driver circuit was a bit curious. Each line from the clock chip drove *two* segments of the four digit display. *Which* segment was being driven depended on which cycle of the 60 cycle clock one was in. I managed to find a 60 Hz signal on the clock board that briefly went low squarely in the middle of one of the valid segment times, and used that plus a 16.7 millisecond delay loop in software to identify which of the two groups of segments my drivers were talking to. Thus, with the information from that clock signal, and with 12 lines of LED driver information, I could get the PIA to "read the clock" by, in effect, literally "looking" at what it showed on its four digit seven segment LED display.

I did a few other tricks: I used 2.2K current limiting resistors and 4.7 volt zener diodes to make current and voltage limiter circuits on each of the LED driver lines, so that their 15 volt levels could be safely fed into 5 volt HCT type TTL chips. I then used two 74HCT245 chips to "multiplex" the data from the clock, allowing me to get away with only eight lines of I/O on the PIA. I used the A port of the 6821 PIA to input this information, leaving the B port for controlling the Darlington tansistors. Thus, with this somewhat "Rube Goldberg" arrangement, I was able to use existing mass-produced technology to get my set-able, battery backed up clock that the project required, and have the information from it assessable to the CoCo that was controlling the rest of the project (Anyone wishing more precise details on how to make a CoCo read a $10.00 Radio Shack LED clock is welcome to contact me thru "68' micros". Depending on your project, there may be a nominal charge for my time and trouble in getting you more detailed software and hardware information. I do have complete schematics and fully tested assembly language software dedicated to this purpose).

When I finally got the project working properly on my "Development System" (my home CoCo 3 system with multi-pak interface, disk controller, and Dennis Kitz 6821 I/O board), I burned an EPROM with the program and checked to make sure that, with an auto-execute ROM pak instead of the disk controller and disk based software, the whole system worked fine. Then I was faced with the problem of bringing over what I developed to a CoCo 2 motherboard.

Space and cost constraints dictated that my controller board could not have a "footprint" any bigger than the size of the actual CoCo 2 mother board, and that no extra cards (no Kitz parallel port card and no ROM pak card) be plugged into the 40 pin system bus port. How to do this???

I added a third PIA chip to my CoCo 2 mother board by piggybacking a third PIA on top of one of the two existing PIA chips in the CoCo. I chose the 6821 (NON keyboard PIA) because it was socketed, allowing me to make up the piggy-back arrangement out of my vast collection of cheap and easily available 6821 PIAs, so that I did not have to mess with the somewhat odd and hard to get keyboard PIA . The piggyback was done as follows:

1) Bend OUT pins 2 thru 17 (all 16 I/O lines), pins 18,19,39, and 40 (the CB1, CB2, CA1, CA2 lines), and pin 23, the *CS2 line of my add-on PIA chip.

2) Position one 6821 on top of another

6821 and solder all down-going pins of the top PIA to the top of pins of the bottom PIA. Note that on the CoCo 2, the CS0 and CS1 lines are both tied active high, and are not used in chip selection.

3) Wire a 20 conductor cable to the 16 data I/O lines of ports A and B of the added PIA chip, and crimp a 20 conductor male dual row connector to that, which provides the exact same pin out and physical port to that on the Dennis Kitz I/O board.

4) Wire a jumper from pin 23 of the added PIA (the *CS2 line) to pin 9 of the LS138 chip of the CoCo 2. Pin 9 of the LS138 is the source of the *SCS line of the CoCo 2. Thus, the added PIA chip is now addressed to $FF50, just as the *SCS line on the Kitz I/O board. Actually, with this arrangement, the added PIA "ghosted" to all four byte address spaces between $FF40 and $FF5C, and so a disk controller could no longer be hooked to the 40 pin system port of the CoCo 2. This is not a problem, for this "controller board" version of the project would not be using a disk controller.

The pins that were "piggybacked" included all system bus data pins, clock pin, read-write pin, power, and ground. Note that this arrangement did not implement the interrupt handling capability of the PIA, but by bending down another pin or two one could have gotten that capability. My project did not require this.

With this task done, I had a spare 16 line I/O port on my controller board that behaved, for my purposes, IDENTICALLY to the I/O board on my development system.

Finally, I had to find a way to add an extra ROM with my auto-execute program to the CoCo 2 mother board. My program was under 2K in size, so I chose to use one of my large collection of spare 2716 chips. I could just as well have used a 2532 or a 2732 chip if I needed 4K of space for the program, and could have even used a 2764 or 27128 if I needed 8 or 15.75K of space.

What I did was piggyback a ZIF socket on top of one of the two 24 pin Basic ROM chips on the CoCo 2 mother board. The rules for wiring a 24 pin ZIF socket to accommodate 24 pin EPROMs are as follows:

1) Bend out pin 20 of the ZIF socket, and jumper it to pin 12 (the *CTS or ROM select line) of the LS138 on the CoCo 2.

2) For a 2716 or a 2532 EPROM, bend out pin 21 of the ZIF socket and jumper that to pin 24 of the ZIF socket. This ties the Vpp line of the EPROM high.

3) Pin 24 of the ZIF socket is, of course, connected to pin 24 of the CoCo ROM, and thus is a source of +5 volts.

4) For a 2732 EPROM, bend out pins 18 and 21 of the ZIF socket, and tie pin 18 of the ZIF socket to ground (pin 14 of the piggybacked socket and original CoCo ROM chip... this ties the Output Enable line low) and connect pin 21 of the ZIF to pin 18 of the CoCo 24 pin ROM (this hooks up the A11 line).

5) If one has 68764, 68766, or 68769 Motorola EPROMs, one can just piggyback all lines except pin 20 of the ZIF socket to either of the CoCo 2 24 pin ROMs. Pin 20 of the ZIF still goes to pin 12 of the LS138 to accomplish addressing the EPROM to the $C000 - $FFFF address range.

6) All of the above is for those models of CoCo 2 that used two 24 pin ROMs on their mother board. The late model CoCo 2B's that used a single 28 pin ROM for BASIC and EXTENDED BASIC are easy to use with 28 pin, 27 series EPROMs. Just piggyback a 28 pin ZIF socket over the 28 pin BASIC ROM, bending out pin 22 of the 28 pin socket and wiring THAT to pin 12 of the LS138 chip. Then JUMPER a wire between pin 28 and pin 1 of the ZIF socket. You now have a ZIF socket that can handle 2764 and 27128 EPROMs nicely. A ZIF socket was used to accommodate easy EPROM changes later. If the EPROM will be pretty much permanently installed for your project, a standard socket can be used. I highly recommend a good quality DOUBLE WIPE socket.

7) Adding a spare cartridge port EPROM to a CoCo 3 would be similar, but for the handling of pin 27. You would bend out pin 22 of the ZIF socket and hook that to pin 12 of the LS138 as before, but would also bend out pin 27 of the ZIF socket and tie it to pin 28. This would allow for the use of 2764 and 27128 EPROMs. You COULD just hook pin 27 of the ZIF to the ROM below it and be able to use 27256 EPROMs too, but that would involve knowing some sneaky details of how the GIME chip registers are used to re-map the BASIC and CARTRIDGE ROMs.

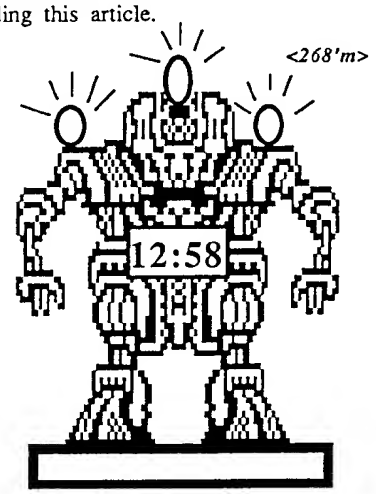
To make the ROM auto execute, I merely jumpered the Q clock line to the *CART line on the CoCo mother board, making the same connection that is normally physically made inside auto-execute game cartridges. This is the "standard" way to make a ROM auto execute. Of course, one must put an ORCC #$50 instruction as the start of one's code at what will be location $C000 in the EPROM, to turn off all interrupts before doing anything else. A sneakier way to achieve auto execution would be to have put the ASCII codes for the letters "DK" as the first two bytes of the EPROM, followed by the code I wanted executed. This is the way the Disk ROM itself is recognized and entered during the initialization of Disk Basic.

With the added, piggybacked PIA and the added, piggybacked ZIF socket addressed to the game cartridge address, I had a "controller board" with a footprint no larger than that of the CoCo 2 mother board, free of any expensive and nasty cables, multi-paks, etc., that provided me with 16 full lines of I/O, space for up to 16K of program, nearly 64K of memory, and that behaved EXACTLY like my development system.

This board fit nicely into the base of the prototype for the toy. I did end up installing a tiny, very quiet fan to cool off the nasty, overly hot power supply that is the single poorest part of Color Computer design. I could have replaced that power supply entirely with a better engineered one either of my own design or off the shelf, and then could probably have done without the fan.

I grant it is rather like trying to swat a fly with a sledge hammer to used a full blown CoCo where the most minimal of controller chips would do. But... it worked, it was economic, and it was easier for me to do than to use a controller chip.

It is my hope that this tale of my recent project will give readers a concrete idea of what I meant all these years when I repeatedly said that the Color Computer makes a GREAT imbedded controller board. I also hope that some readers can get ideas about how to use THEIR CoCos to accomplish "real world tasks" after reading this article.



<268'm>

The Industrial OS-9 User

Ed Gresick

The how and why of Microware's marketing of OS-9.

Editor's Note: *While perusing the Delphi OS-9 forum, I came across the following answer to a simple question asked by "THETAURUS" (Delphi user name)- why is there no OS-9 graphics standard? Mr. Gresick's answered not only that question in his answer, but also shed some light on several items about Microware and OS-9 that many people are not aware of. With the permission of Ed and THETAURUS, I present the following.*

Basic Marketing Strategy

With a couple of exceptions, MW does not offer OS-9 to the end user. OS-9 is sold to the hardware manufacturer (OEM). Depending on the licensing agreement, this will include the kernel, some or all of the file managers, the utilities, assembler, linker, C-Compiler, etc. It does not include drivers or descriptors although MW does provide _unsupported_ sample drivers and descriptors for some of the more popular I/O chips. It is up to the OEM to write his own drivers and descriptors. This approach was taken by MW topermit each OEM to optimize his hardware for his market (MW will write drivers for OEMs under a separate contract). Thus, the capabilities of aspecific hardware package from an OEM are determined primarily by and are the responsibility of the OEM. The exceptions I mentioned above are OS-9000 for the 386/486 and several Motorola VME boards.

I don't have any actual figures but I believe that the bulk of OS-9 salesare for rommed, diskless applications. If they include any I/O it will be for data collection, processing and control; they will not include any human user interface.

The next level might by something like CD-I. The OS is contained in a ROM and it reads and controls a optical disk. Output is to a TV set and other input limited to a few commands.

The systems most people in the hobby arena (specifically, the Delphi OS-9 forum) are familar with are disk based. The bootromcontains only enough info to find the bootfile on the disk, load it and switch operation to OS-9 (The CoCo version uses software instead of a bootrom).

Up to a few years ago, OS-9 systems which required user interface used text terminals. These were mostly development platforms although some were used for business applications. Of course, there were always a few hobbyists. The only hardware available offering graphics was the CoCo. While MW did write much of the code, this was done under contract to Tandy and Tandy owns the rights.

Why no OS-9 graphics standard?

More recently, we've seen customers who want graphics capabilities. Several of the OEMs providing VME equipment wrote drivers capable of driving graphics terminals. Others have designed graphics boards for their equipment. Each OEM has selected the graphics chip he felt best suited his market's needs.

Since the graphics chips are different, each driver is substantially different. And each OEM has written his own equivalent of CGFX.1 libraries and calls.

The nearest thing to a GFX standard is MW's RAVE. Currently, RAVE supports the Vigra MMI-100, MMI-250 and Matrox VIP-640 graphics processors. So far, RAVE seems to be accepted only for a few dedicated apps. But, RAVE has been selected for Bell Atlantic's VOD system, which means it may catch on more in the future.

Then, there are the various windowing packages; X-WINDOWS, G-WINDOWS, K-WINDOWS and MGR. X-WINDOWS is not practical for the 'lesser' microprocessors. MW recommends that a 68040 be used. G-WINDOWS has probably received the greatest acceptance by OEMs and users. K-WINDOWS, so far, is limited to one hardware platform and MGR's use seems to be limited to a few European companies.

For the SYSTEM's IV and V Computers, Delmar has settled on video cards using theTSENG LABS ET4000/ ET4000w32i graphics chips. (TSENG LABS is also designing the Vigra chip.) Our driver for these boards supports text modes from 40 x 25 to 132 x 44 and graphics modes up to 1024 x 768 x 256. A graphics C library is available which is a sub-set of Microsoft's Quick C. While we sell and support G-WINDOWS, G-WINDOWS is not necessary for gfx on either computer.

Several people have written gfx programs. There is Bob Hollman's port of TETRIS. Very nice gfx and action. Dave Proctor has written an excellent 'flic' viewer. Several others have written specific programs using the SYSTEMs IV & V gfx capabilities.

The way things stand now, even if you get a machine that supports gfx, software you write probably won't be portable to other hardware. Of course, if you're doing this for your own amusement, who cares?

Gfx compatibility across hardware platforms is one of the reasons for the various windowing systems being offered. Gfx written under a given windowing system will (or should) run on any hardware supporting that windowing package.

< 268'm >

Operating System - Nine

continued from page 15

Wcreate's only feature over display is that it accepts decimal numbers. Personally, I don't think it's worth 8K for that. 80=50, 40=25, 24=18. Done.

Xmode isn't used that often, and there are two competing versions. (stock and sacia). Leave it separate so either one can be loaded. I renamed sacia's xmode smode and don't load either one at boot. Some programs are hardcoded to load xmode and use the old variable names- these run fine. When I need to modify my serial ports, I can explicitly load smode (which is still internally identified as xmode).

I sort of wanted to get to some apps this time, but if Frank has to drop ANOTHER point size to fit my mess in somebody might shoot one of us- so this is the end for now.

Before I go, a word about the various pd and wholly owned programs mentioned here. If you don't have a modem or don't have a decent on-line source for something, I'll do a little dsaving on your behalf. Rather than split things up by columns, I'd rather go by the diskful. I figure $5 a disk is about a wash. I'll use the biggest format your machine supports- up to 3.5 inch 720K (sorry, don't have a 80 track 5.25 or 1.2/1.4 meg). Whats more, the contents are pretty much up to you- a bit of article #2 and some of #4 plus a few things from #6.... whatever fits on the format. I suppose the thing to do is start with what is needed most, then on down to the 'well, might as well add this....' range.

As time passes, this offer will get more useful. As volume increases, I will quit bringing it up. So you, Gentle Reader, will have to remember it was made. The first person who mails me the name of the author who popularized that phrase can pick any late model regular stock Tandy (CoCo) game they want.

< 268'm >

Operating System -Nine

NOTE: Though based on CoCo OS-9, this column is good for OS-9/68000 beginners also!

Rick Ulland

Disk organization, multi-user notes, efficient RAM use, and some great utilities.

It's amazing how many words one can write about an Operating System before getting to the point- a fulltext search for "user data" shows no hits. Time to spin my centricities thataway.

The 'power' of a computer can be summed up in the amount of data it is capable of handling in a given amount of time, it's thruput. As many CoCoists have discovered, a CoCo's thruput isn't that bad, in fact, it seems to stray up towards the 286 range at times. This isn't enough to do fancy graphics in real time- just to many bytes to shove down the buss. On the other hand, that's several times what's needed for a user to preform actual work- humans don't create data very fast, even an 8 bit buss can keep up with 2 or 3 of 'em.

Anyone who has grown up with a CoCo system using basic knows that it was based on the then current technology (early 80s) of the cassette recorder, with extensions to use the current hot stuff, a floppy drive. The operating system built into the machine represents a level of complexity roughly equal to the supplied hardware.

Disk Basic's compatibility problems with hard drives aren't due to some evil plan on behalf of Tandy's Master Agenda- the assumption was made that a few floppies was as far as a CoCo would go. OS9 (in it's desktop incarnation) has the opposite problem. It does what it does, hardware be hanged. The CoCo has enough trouble keeping up, and OS9 has compatibility problems with floppy drives beyond those caused by cheap halting controllers. This makes the expense of a hard drive easier to justify- besides the obvious speed and capacity advantages, the user is moving into the environment the OpSys expects, and the surprises are usually pleasant. All that's missing is a tape backup (SBF) driver for 6x09 OS9. (Wanna be a rock and roll star and have folks kiss your feet at CoCoFests? Write 6x09 SBF.)

Regardless of your actual storage capabilities, OS9 demands it's space. The floppy solution outlined last month allows a pretty complete system to squeeze into a 2 floppy machine. Or chuck the whole thing and spend your fishin' money on that hard drive.

If programming is your bent, there are additions to the system directories- /dd/DEFS and /dd/LIB. Once the system files are nestled away on /dd, we get to the good stuff-actual user data. (There! I said it!)

With floppies, organization doesn't get that deep. It's still helpful to look at how larger devices are organized- you might buy one someday. I use three main data branches to keep my intellect intact:

/SRC is split into billions of subdirectories. Branches for different languages fan into little directorilets to contain separate projects and versions. SRC has it's own CMDS directory for strange execs, alternate runb merges, etc. The important thought is, src is untested work which should be quarantined..

/USR. This is where the work get done. With an array of sensible subdirs like DESK,GAMES,GFX,TCOM..... an inexperienced user can be told "Just click USR, then anything...."

/CONECT is my company, which has tax implications and deserves it's own branch. You no doubt have a similar all consuming advocation that deserves it's own- that novel or all of those not ready for prime time gfx. (See multiuser.)

Beyond this, you can of course add anything you want. In fair trade, the machine can add anything IT wants- don't be surprised to see strange files or directories pop up in root,sys, cmds, or even see a programmer cheat and put stuff in "your" user data directories!

The general rule is, when in doubt, leave it. Remembering what your different programs use for 'scratch' file names can be helpful- you can be certain many errors you see are accompanied by a strangely named scratch file someplace..... Often a blown save or badly timed power sag can be remedied, at least in part.

Multiuser Systems

As with almost any computer, the easy way to allow others to bang around is with a BBS program- but this is far from the only option. One of the neat things OS9 allows is giving a caller a normal shell- anyone who is old enough to remember dialup timeshare systems will get a kick out of setting their OS9 system up as the 'mainframe'.

Although this does make an easy to use BBS (if your users know OS9), there are a few things to do before allowing John Public his own process number. Mostly, this is a matter of removing public permissions from files or entire directories, using attr (Don't forget to make attr itself private!).

Beyond that, the level of security on your system depends on what you allow users to do. If they have access to any languages or assemblers, hackers can always slip in easily enough, although letting the average joe play with your basic09 or c may not immediately be a problem.

Strangely enough, this is not in violation of your Tandy software license- the software is still running on ONE computer- Tandy explicitly allowed terminals if the software supported it, most OS9 software does by default. 3rd party software depends on the 3rd party.

I have seen systems which allowed a user to upload an executable to (user) CMDS then run it. This is a terrifically stupid idea- allowing folks to debug their virus at home, then phone it over and install it in a single session.

Even after all this, there is a backdoor in. The stock shell will let anybody do raw disk reads and writes. A little diddling with /sys/password and any one can make themselves superuser for a day. Shell+ strips out the @ character needed to pull this off- (unless you already are superuser). Use it!

Using your RAM efficiently

I've received quite a few letters concerning memory management and similar concerns, so we are wading in there again.

Once an OS9 system passes about 256K, it starts getting good. The reason is simple, OS9 itself runs around 200K unless special efforts are made to squash it down. 128K machines can barely get up a shell before mem errors start. I had an aggravating reintroduction to the marvels of 128K recently, while setting up a hard disk for a customer. (2 rooms full of CoCo stuff and the only machine besides this one that was up was a 128K 6309. Go figure.) You have my condolences. 512K is a decent size, and 2 megs is astounding. You'd think with all that RAM there is no need to worry. You'd be wrong. The rub is, the 6x09 can't handle more than 64K at a time. While RAM can increase the number of these 64K chunks available, they don't get any bigger.

There are ways- the Sierra games (and flight simulator) use a virtual RAM driver to flip blocks of RAM in and out, effectively gaining use of more RAM. Bruce Isted has written a public domain package to do the same thing. Besides giving hackers a way to play, VRN can replace both the AGIVIRQDr and FT drivers, easing up yet another form of RAM cram- OS9Boot also has to fit in 64K!

Although the above programs do the job, they aren't really well suited for day to day use. Not many have them so few write programs to use them. You can of course write your own, but others will have a bear of a time getting it running- just what OS9 doesn't need.

Anyone who is interested in the internal workings of all of this should scrounge up a copy of Kevin Darlings utility package KDUtil(pd, available everywhere). A few of these are real jewels-

Smap displays a graphic of system memory use, and fills a serious hole in the toolbox. With smap, one need never be surprised with a 237 error- they are easy to see coming as the block gets fuller, and fuller.....(Note that under OS9 'System RAM' isn't the amount on your system, but the amount the Operating System has- 64K at best. Flat out of RAM is a 207 error)

The neatest util in this package has to be **pmap**. PMap lists all of the current processes, and the actual physical RAM blocks each one uses. Watching pmap for a while is more enlightening than pages of manual. You can watch data blocks building up from the bottom, program descending from above, and guess how long till they collide in the middle. Show your DOS friends how OS9 will map the same block of code (say shell) into as many processes as need it- when is 512K=576K? When nine shells are running!

Each line in pmap has 8 slots, since normally you only have eight blocks per process. Even the smallest utility needs some data space, so the smallest a process can normally be is 2 blocks. Lots of things allow the user to increase the number of data blocks to fill up all 8- thats what copy ....#56K is allabout.. What if you need more?

A workaround you may already be using is a pipe. The example I generally use is dsave. DSave uses copy, and makdir, and shell.....and it's pretty big itself. Cramming all this in one process wouldn't leave much data space for anyone, and copy likes large chunks of data space. So dsave runs by itself, and commands are piped to a separate process- doubles the space.

Another way is to split programs into logical sections. My word processor (DynaStar) nas a decent size buffer- only because the printer code is a separate program. The speller makes a third unit, and all benefit from the extra room.

A more esoteric fix is the data module. Although these aren't a feature of the stock shell, they are included in shell+. Although not universal, shell+ is pretty popular and isn't that hard to install, avoiding the problems VRN has. Oh yah- data modules. As mentioned above, OS9 has no problems with mapping the same program code into multiple processes. Pulling the same trick with data blocks gives any number of processes access to them- the data module becomes the common meeting ground for all of these otherwise unrelated processes.

Total RAM would seem to be an absolute limit, but our DAT memory system clouds the issue. Since memory is handed out in blocks, anything loaded off disk has to take a whole block for program. This is NOT a good thing- piddly little 200 byte utilities hogging 16K each (block of program, block of data) will rapidly eat up your RAM, and the need to keep unlinking and reloading will rapidly eat up your patience.

This is the reason for merged files. Since OS9 is loading by the file, not by the utility, it makes sense to form each file into block sized chunks. Actually, 256 bytes (one page) smaller than a block (7936 bytes total) seems to be about the limit before it spills over into another block.

However, once a group of utilities is loaded as a unit, it remains a unit. Think of the result as one big multifunction utility. Only the first module gets a link count, and the remainder tag along. Unlinking that first util to 0 drops the whole ball of wax. It's important to keep this in mind when making up merged files, and start them with a util thats not likely to be unlinked.

For an example, I'm going to trot out dsave again. With most utils in merged files, I didn't see a need to keep individual copies on the disk as well. Worked fine until I ran dsave, which tried to load copy. It wasn't there, so errored out- but dsave runs with error abort turned off, and copy is in RAM anyway. So far so good. At the end of the run, dsave unlinks copy. This part worked all to well, since my alphabetized merged file had copy first. The whole thing disappeared- and mdir was one of the things that went. I couldn't even see what had happened! If you cheat and peek at the file list, you'll see copy is now safely in the middle!

Another problem to avoid is overly big merged files. When OS9 maps one into a process space, it has to map the whole thing in. This is why you want to keep the size down to a block or two. Looking at the listings again, you'll see I put copy in the 1 block sutil file, not the 2 block dboot file where it logically belongs. Allows another data block for the memory hungry copy command.

Some things aren't worth merging. Dsave (hehe- #3) and os9gen are to big to mess with. Some merges are more for the users convenience than the systems- many recommend merging runb, gfx2, syscall, and inkeyS together. The same thing can be done to Basic09, at the loss of some data space. The extra modules don't appear in B09 workspace at first, but magically appear on cue- with no further loads from disk. I have a second copy (TBasic09) without the added stuff for those really big programs.

While we are here, I discovered a neat feature recently. Properly set up, the newer gshells will call runb for you if a packed B09 file is clicked on. Well, runb got clobbered in a recent hard drive crash. Whats an Operating System to do? Why, use Basic09 itself, of course. The only way I knew this had happened was after quitting a program. Instead of disappearing she went to the Basic09 prompt, since B09 doesn't die when a program ends like runb does.

Shell+ was mentioned above. This shell adds a lot, but there is a cost. Tandy did their own file merging so the original shell has vital utilities like echo and load merged with it. Keeping the bigger and better shell+ small enough (remember, shell gets mapped everyplace- we sure don't want it hogging extra blocks) requires most of these be removed. The docs suggest shell, load, list, merge, del and dir- good choices if you think about it.

Keeping shell+ down to a block can cause some strange problems if your cmds directory has been stripped down to save disk space. For instance, startup might appear to not run at all! Why? Because startup aborts on error. Any startup that still begins 'echo Welcome to OS9 on....' is going to abort right there, since echo is no longer merged into shell. The error code goes to the shell thats running startup, but this shell dies with startup- you don't even see the error.

The solution is to load your merged files right away on the first few lines of startup. At least the sutil file- then you can use echo to print status reports, display to set up windows, and such.

File Lists

To make up these files, chd to cmds and use merge file1 file2 file3 >/utilfile (thats merge space, filenames separated with spaces, >/putemhere). Then reset the execute attribute- attr utilfile e pe for single user systems.

My shell+ boot looks like this:

'shell' (one block) - shell+, load, list, merge, del, dir

'sutil' (one block) - mfree, date, copy, deiniz, display, echo, iniz, link, procs, rename, setime, tmode, unlink

'dboot' (two blocks) - attr, deldir, edit, format, free, ident, makdir, backup, cobbler

separate - dsave, os9gen, xmode

missing - pwd, pxd, wcreate

Notes:

Some programs use pwd or pxd, so you might want to keep a disk copy in case it comes up. Users can run the built in shell+ equivalent .pwd and .pxd, but without hacking programs don't know these are available.

continued on page 13

Rick Ulland can be reached in care of this magazine, via electronic mail (Delphi: rickuland; Internet: rickuland@delphi.com), or U.S. mail at 449 South 90th, West Allis, WI 53214. Feel free to send questions or comments, but include an SASE if expecting a personal reply. Items sent to the magazine will be considered for publication.

OS-9/OSK Answers!

Joel Mathew Hegberg

Detect Child Process, OS-9/68K Data Modules, Bells & Whistles for K-Windows

Welcome back! In my last issue I left off giving some C source code for playing a digital sound under K-Windows. This time, I'm pleased to write that Ted Jaeger has sent me some BASIC source code ("play.bas" — listing #1) to demonstrate playing .IFF sound files under K-Windows. Thanks, Ted! Play.bas requires the two subroutines "htoi.bas" and "itoh.bas" (listings #2 & #3, respectively) to help determine the IFF sound file header information.

One advantage to using C over BASIC for a 'play' function is memory allocation.Using C, a programmer can dynamically allocate memory to store a sound in memory (see my column in the last issue for an example). With BASIC, you're pretty much stuck with the amount of memory you received at run-time. Worse, under OSK, the user is the one that has to remember to request additional memory, if needed. Ted handles this BASIC shortfall very nicely by using the windowing system to dynamically allocate the memory needed for the sound to reside in. By using K-Window's GetBlk function ($1b $2c) and some strategic SysCall's, "play.bas" defines a get/put buffer, determines its memory address, reads in the digital sound data, starts the sound sample playing, and then releases the buffer memory back to the system.

One interesting note, under MM/1 K-Windows, digital sound is controlled by a DMA sound chip which takes care of all the sound output, thus freeing up the system's CPU for other tasks at hand. When Ted's program exits, the sound sample is still proba bly not finished playing (unless it's extremely short — as in less than a second in length), yet it's memory is deallocated when the program exits. Will this cause a problem? Not really. The worst thing that can happen under this circumstance would be for another process to allocate the same memory the sound sample is occupying, and write over the sound sample before it's finished playing. In that instance, the sound data would be corrupted, and you would hear garbage for the remaining play duration. In my experience, this is a rare case scenario — but rest assured you can induce such a occurrence if your heart is set on it!

Listing #1 — "play.bas"

```
==========
PROCEDURE play
(* Subroutine to play brief digitized
(* sounds saved  in monophonic iff
(* format . Call play passing complete
(* path name of the sound file e.g.,
(*  /dd/snds/ding.iff.
(* If packed run with new RunB
(* (data size=8000 bytes)
(* December 15, 1993

PARAM filename:STRING[20]

(* register mask for system calls
TYPE registers=d(8),a(8),pc: INTE-
GER
DIM regs:registers
DIM callcode:INTEGER

(* avoid BGFX by setting up variables
(* to PUT graphics commands
DIM openbuf(12):BYTE
openbuf(1)=$1b
openbuf(2)=$2c
DIM kilbuf(4):BYTE
kilbuf(1)=$1b
kilbuf(2)=$2a

(* array for first 34 bytes of sound file
(* header -  we'll find length and
(*  frequency info here
DIM header(34):BYTE

(* variables for integer to hex and
(* hex to integer conversions
DIM x,y,z:BYTE
DIM x$,y$,z$:STRING[2]
DIM answer:INTEGER
DIM answer$:STRING[6]
DIM hex:STRING[2]

(* sound file buffer data
DIM bufsize:INTEGER
DIM buflocate:INTEGER
DIM pixels:INTEGER
(* process id needed so
DIM id:INTEGER
DIM group:BYTE

DIM path:INTEGER

ON ERROR GOTO 1000

(* get the process id
(* for group number of buffer
callcode=$0c
RUN syscall(callcode,regs)
id=regs.d(1)
group=id

(* load first 34 bytes of sound file
header
OPEN #path,filename
GET #path,header
CLOSE #path

(* get the buffer size of sound file
x=header(6)
y=header(7)
z=header(8)
RUN itoh(x,hex)
x$=hex
RUN itoh(y,hex)
y$=hex
RUN itoh(z,hex)
z$=hex
answer$=x$+y$+z$
RUN htoi(answer$,answer)
bufsize=answer

(* if sound file too large abort
IF bufsize>53248 THEN
END "Sound file too large"
ENDIF

(* get sampling frequency of sound file
x=header(33)
y=header(34)
RUN itoh(x,hex)
x$=hex
RUN itoh(y,hex)
y$=hex
answer$=x$+y$
RUN htoi(answer$,answer)
freq=answer

(* open a buffer to hold sound file
(* size of buffer determined in offset
(* 9, 10, 11, 12 of openbuf array
(* x dimension of buffer fixed at 512
(* y dimension of buffer varies to allow
(* proper size  for holding sound file
(* size calculated by multiplying # of
(* bytes in buffer X 2.  this indicates # of
(* needed pixels in buffer because in
(* Type 0 screen there are 2 pixels per
(* byte. pixels=bufsize*2.  since 512
(* pixels per row we can divide total
(* pixels  by 512 to determine how many
(* rows are needed in open buffer call
y=pixels/512

(* cant exceed max  # of rows on screen
IF y>208 THEN
y=208
ENDIF

openbuf(3)=group
openbuf(4)=1
openbuf(5)=0
openbuf(6)=0
openbuf(7)=0
```

KiX/20...

32 bits of affordable excellence

32 bit Performance at 16 bit cost!

NEW LOW COST 32 BIT COMPUTER FOR OSK!

The KiX\20 has the same 32 bit expansion bus as the KiX\30 and uses the same expansion boards. This includes the Ultra fast Multi Graphics Adaptor video board. The 32 bit MGA video board adds high end graphics to the low cost KiX\20. This combination provides the end user with high performance workstation capabilities at home computer prices. *If you are looking for an OSK machine that provides super fast video and low cost, then this is it.*

The KiX\20 specs:

- ►25 Mhz 68020 CPU
- ►1, 2, 4, 8 or 16 Meg of RAM on the motherboard using SIMM memory.
- ►On board SCSI controller (same as KiX\30)
- ►Dual density floppy controller. Supports 360K to 1.4 Meg floppies.
- ►8K Battery Backed Static RAM.
- ►4 on board full serial ports (same as KiX\30)
- ►Battery backed Real Time Clock
- ►Sound port
- ►1 Parallel printer port
- ►Full user manual with schematics
- ►Autoboot from floppy or hard disk included.
- ►4 Built in I/O expansion bus.
- ►1 KiX\30 32 bit Expansion bus connector
- ►4 layer PCB
- ►Small 8.7 in by 8.7 in. size drops in any PC case

includes...

- ►PROFESSIONAL OS9/68000
- ►MICROWARE K&R C COMPILER
- ►MICROWARE BASIC!
- ►MICROWARE MANUALS
- ►Run as a terminal system now, then add a video board later.

The bus runs at full CPU speed. The bus is not a limiting factor like the slow AT/ISA bus used on most PCs available today. (The AT/ISA bus is limited to 8Mhz and is only 16 bits.)

The performance gain of this system over current systems is dramatic.

In today's economy, cost is a major factor. This system is designed to start at only 699.95. That includes Professional OS9/68000, Microware K&R C compiler, Microware BASIC, deguggers and tons of utilities.

You can start out with a terminal based system and add the MGA video board later.

The introductory price for the KiX\20 is:

Only $699.95!

And that includes...
OS9/68000, C and BASIC!
Order today!

FRANK HOGG LABORATORY, INC.

204 Windmere Road Syracuse, NY 13205 *Fax:* 315/469-8537 *Telephone* 315/469-7364

KiX Systems

KiX\20 SALE

Terminal Systems

KiX\20 MB w/OSK	699.95
Case/PS/Floppy/Cables	249.95
Price seperately	949.90
Save	-50.00
ONLY	**899.95**

Video Systems

KiX\20 MB w/OSK	699.95
MGA Fast 32 Bit Video	450.00
G-Windows GUI	275.00
Case/PS/Floppy/Cables	249.95
Price seperately	1674.90
Save	-274.95
ONLY	**1399.95**

KiX\30 SALE

KiX\30 MB w/OSK	*1599.95
Case/PS/Floppy/Cables	249.95
Price seperately	1849.90
Save	-50.00
ONLY	**1799.95**

* 16 MHz version
Add $100/200 for 25 MHz /33 MHz

KiX\30 MB w/OSK	*1599.95
MGA Fast 32 Bit Video	450.00
G-Windows GUI	275.00
Case/PS/Floppy/Cables	249.95
Price seperately	2574.90
Save	-274.95
ONLY	**2299.95**

* 16 MHz version
Add $100/200 for 25 MHz /33 MHz

Hard Drives

Add a Hard Drive to any of the above systems. All hard drives come pre loaded with the latest software ready to use.

Quantum 52 Meg	149.95
Quantum 170 Meg	299.95
Maxtor 245 Meg	375.00
Maxtor 350 Meg	525.00
Quantum 500-+ Meg	Call

Note: All systems require some assembly. No soldering. Takes about 1 hour. All components are pre tested and guaranteed for 1 year. All are shipped with Zero K RAM. Call for latest prices on SIMM DRAM. Uses standard Mac (8 bit) or PC (9 Bit) SIMMs.

Software

35% OFF W/SYSTEM

DynaStar 140.00
Word Processor with the easy to use and familiar user interface. The standby used by many OS9 users.

VED Word processor 60.00

Fast BackUp 50.00
Backups by file list using compression.

Sculptor v1.14 200.00
Older version at greatly reduced cost.

KEEPER Business Software 299.95

MM/1 VERSION NOW SHIPPING!

G-Windows

Designed Specifically for OS-9 Users

The *G-Windows* windowing software goes far beyond anything ever offered for the OS-9 operating system. Its modular structure, multi-tasking capabilities, and unique way of seamlessly interfacing with the user's application program make it a breakthrough in the emerging technology of graphical user interfaces.

G-Windows pricing

G-Windows for KiX	275.00*
G-Windows for MM/1	200.00
G-Windows for OS9000	275.00

Includes: Window manager and required modules. Desktop manager and related files. Fonts and image modules. Basic set of gadgets used by many utilities. GIF viewer. Manual: "Using G-WINDOWS/Desktop Manager" Easy setup information.

*KiX versions are packed with the MGA video board at reduced cost.

G-Windows Development

Develop software for G-Windows.
OS9/68000 Dev System 300.00

FRANK HOGG LABORATORY, INC.

204 Windmere Road Syracuse, NY 13205 *Fax:* 315/469-8537 *Telephone* 315/469-7364

```
openbuf(8)=0
openbuf(9)=2
openbuf(10)=0
openbuf(11)=0
openbuf(12)=y
PUT #1,openbuf

(* map the buffer
callcode=$8d
regs.d(1)=1
regs.d(2)=$a9
regs.d(3)=group*256+1
RUN syscall(callcode,regs)
buflocate=regs.a(1)

(* open and read in entire sound file
OPEN #path,filename
(* I$Read
callcode=$89
regs.d(1)=path
regs.d(2)=bufsize
regs.a(1)=buflocate
RUN syscall(callcode,regs)
CLOSE #path

(* play sound file
callcode=$8e
regs.d(1)=1
regs.d(2)=$97
regs.d(3)=bufsize
regs.d(4)=freq/2
regs.d(5)=0
regs.d(6)=0
regs.d(7)=0
regs.a(1)=buflocate
RUN syscall(callcode,regs)

(* kill the buffer holding the sound file
(* VERY important because sound
(* buffers are large and RUNB limits
(* data memory  to 8000 bytes
kilbuf(3)=group
kilbuf(4)=1
PUT #1,kilbuf

END
1000 (* report error
errnum=ERR
PRINT "got error "; errnum
```

Listing #2 — "htoi.bas"

```
========================
PROCEDURE htoi
(* convert a 6 digit or smaller hex value
(*  to integer hex value received as a
(* string in hex$
(* December 15, 1993

PARAM hex$:STRING[6]
PARAM answer:INTEGER

DIM digit(6):INTEGER
DIM l$:STRING[1]
DIM l:INTEGER

DIM count:INTEGER

(* examine digits in hex$ individually
FOR count=1 TO LEN(hex$)
l$=MID$(hex$,count,1)

(* and convert them to integer format
IF l$="a" OR l$="A" THEN
l=10
ELSE
IF l$="b" OR l$="B" THEN
l=11
ELSE
IF l$="c" OR l$="C" THEN
l=12
ELSE
IF l$="d" OR l$="D" THEN
l=13
ELSE
IF l$="e" OR l$="E" THEN
l=14
ELSE
IF l$="f" OR l$="F" THEN
l=15
ELSE
l=VAL(l$)
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

(* now get the value of each digit
digit(count)=l*16^(LEN(hex$)-count)

NEXT count

(* now find total of converted hex values
answer=0
FOR count=1 TO LEN(hex$)
answer=answer+digit(count)
NEXT count

END
```

Listing #3 — "itoh.bas"

```
========================
PROCEDURE itoh
(* Convert a value < 256 to hex
(* December 15, 1993

PARAM value:BYTE
PARAM answer$:STRING[2]

DIM x:INTEGER
DIM x$:STRING[1]
DIM digit1,digit2:STRING[1]

(* get digit in 16s place first
x=value/16
GOSUB 10
digit1=x$
(* now get digit in 1s place
x=MOD(value,16)
GOSUB 10
digit2=x$

(* now add the two digits for hex value
answer$=digit1+digit2

END

10 (* get one hex digit of the answer
IF x=15 THEN
x$="f"
ELSE
IF x=14 THEN
x$="e"
ELSE
IF x=13 THEN
x$="d"
ELSE
IF x=12 THEN
x$="c"
ELSE
IF x=11 THEN
x$="b"
ELSE
IF x=10 THEN
x$="a"
ELSE
x$=STR$(x)
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

RETURN
```

Ted has also sent me some source for creating a fancy shell window from within BASIC (listing #4). For those who prefer to program under C, I've ported it for you and included the source (listing #5). The window which is created for the shell is NOT an overlay window, but rather is an independent window, which can be moved, resized, or pushed back. To close up the shell window, just hit the 'Esc' key. Ted stayed away from direct BGFX calls for his BASIC source, so I stayed away from the CGFX library calls in my C version, just in case anyone thought you absolutely had to use CGFX library calls to do K-Windows screen functions.

Listing #4 — "popup.bas"

```
========================
PROCEDURE popup
(* place a sizeable, relocatable, push-
(* back  window on the current Type 0
```

```
(* screen press ESC key to exit
(* December 16, 1993

(* register mask for system calls
TYPE registers=d(8),a(8):INTEGER
DIM regs:registers

(* variables to make and add features to
(* the new window. let's avoid forking
(* shells or calling BGFX so we'll set up
(* to PUT the commands
DIM dwset(9):BYTE
DIM frame(3):BYTE
DIM move(21):BYTE
DIM shape(22):BYTE
DIM push(21):BYTE

(* variables to dup paths
DIM oldpath(3):INTEGER
DIM newpath:INTEGER
DIM path:INTEGER
DIM I_Dup:INTEGER
I_Dup=$82

DIM errnum:INTEGER

ON ERROR GOTO 100

(* open the path to new window
OPEN #newpath,"/w"

(* set it up to display on same screen
(* with white foreground and green
background
dwset(1)=$1b
dwset(2)=$20
dwset(3)=$ff
dwset(4)=$08
dwset(5)=$0a
dwset(6)=$40
dwset(7)=$0c
dwset(8)=$ff
dwset(9)=$04
PUT #newpath,dwset

(* give it fancy frame
frame(1)=$1e
frame(2)=$22
frame(3)=$05
PUT #newpath,frame

(* click upper left corner to move
move(1)=$1e
move(2)=$21
move(3)=$00
move(4)=$02
move(5)=0
move(6)=0
move(7)=0
move(8)=0
move(9)=0
move(10)=0
move(11)=0
move(12)=$10
move(13)=0
move(14)=$08
move(15)=0
move(16)=1
move(17)=$80
move(18)=0
move(19)=0
move(20)=0
move(21)=0
PUT #newpath,move

(* click lower right corner to scale
shape(1)=$1e
shape(2)=$21
shape(3)=$00
shape(4)=$03
shape(5)=$00
shape(6)=$00
shape(7)=$ff
shape(8)=$f0
shape(9)=$ff
shape(10)=$f8
shape(11)=$00
shape(12)=$10
shape(13)=$00
shape(14)=$08
shape(15)=$00
shape(16)=$02
shape(17)=$81
shape(18)=$0d
shape(19)=0
shape(20)=0
shape(21)=0
shape(22)=0
PUT #newpath,shape

(* click in upper right corner to push
back window
push(1)=$1e
push(2)=$21
push(3)=$00
push(4)=$04
push(5)=$00
push(6)=$00
push(7)=$ff
push(8)=$f8
push(9)=$00
push(10)=$00
push(11)=$00
push(12)=$08
push(13)=$00
push(14)=$08
push(15)=$00
push(16)=$01
push(17)=$82
push(18)=0
push(19)=0
push(20)=0
push(21)=0
PUT #newpath,push

(* select new window
PRINT #newpath,CHR$($1b);
CHR$($21);

(* dup the paths
BASE 0
FOR path=0 TO 2
regs.d(0)=path
RUN syscall(I_Dup,regs)
oldpath(path)=regs.d(0)
CLOSE #path
regs.d(0)=newpath
RUN syscall(I_Dup,regs)
NEXT path

(* give new window a shell
SHELL ""

(* return original paths
FOR path=0 TO 2
CLOSE #path
regs.d(0)=oldpath(path)
RUN syscall(I_Dup,regs)
CLOSE #oldpath(path)
NEXT path

(* select original window
PRINT #0,CHR$($1b); CHR$($21);

CLOSE #newpath

END

100 (* report error
errnum=ERR
PRINT "Got error "; errnum
```

Listing #5 — "popup.c"

```
#include <stdio.h>
#include <modes.h>
#include <errno.h>

main()
{
    int save_path[3],wpath,t;

   /* Open a path to a new window. */
  wpath=open("w",S_IREAD|S_IWRITE);
  if (wpath==-1)
          {
      fprintf(stderr,"Cannot open path to
new window!\n");
                  exit(errno);
          }

    /* Set it up on same screen, white
on green. */

slow_write(wpath,"\x1b\x20\xff\x08\x0a\
x40\x0c\xff\x04",9);

          /* Give it a fancy frame. */
          write(wpath,"\x1e\x22\x05",3);
```

```
   /* Click upper-left corner to move */

write(wpath,"\x1e\x21\x00\x02\x00\x00\
x00\x00\x00\x00\x00",11);

write(wpath,"\x10\x00\x08\x00\x01\x80\x00
\x00\x00\x00",10);

   /* Click lower-right corner to scale */

write(wpath,"\x1e\x21\x00\x03\x00\x00
\xff\xf0\xff\xf8\x00",11);

write(wpath,"\x10\x00\x08\x00\x02\x81
\x0d\x00\x00\x00\x00",11);

   /* Click upper-right corner to push
back. */

write(wpath,"\x1e\x21\x00\x04\x00\x00
\xff\xf8\x00\x00\x00",11);

write(wpath,"\x08\x00\x08\x00\x01\x82
\x00\x00\x00\x00",10);

   /* Select the new window. */
     write(wpath,"\x1b\x21",2);

    for (t=0;t<=2;t++)
    {
         save_path[t]=dup(t);
         close(t);
         dup(wpath);
    }

      system("shell\n");

    for (t=0;t<=2;t++)
    {
         close(t);
         dup(save_path[t]);
         close(save_path[t]);
    }

     close(wpath);
}

/* slow_write outputs 'data' one byte */
/* at a time.  Needed due to a problem*/
/* with DWSet codes. */
slow_write(path,data,size)
int  path;
char *data;
int  size;
{
   int  t;
    char *c;


for(t=0,c=data;t<size;t++,c++)
write(path,c,1);
}
```

Before I logout, I have a couple corrections to make. One correction regards my last column (01 FEB). Colin McKay pointed out that for the listing "playbell.c", an additional INCLUDE statement is needed. After the #include <stdio.h> line, the following line was omitted:

#Include <types.h>

My thanks to Colin for pointing this out to me. I've modified my "MACHINE/regs.h" file so it doesn't require the "types.h" header file, which was the cause of the omission.

The second correction is for the article in issue number 4 (15 DEC). A small error was made in Listing #2, it should read like this:

Listing #2:

```
#include <module.h>
main()
{
   char *dataptr,*mod;
   /* Pointer to whatever data-type */
   /* to be stored */
   mod_exec mp;

     mp=_mkdata_module(modname,
                datasize,attrevs,perms);
   mod=(char *)mp;
    dataptr=(char *)(mod+mp>_mexec);
   /* data pointer */
}
```

Best wishes to everyone, and as always, feel free to send in any questions or answers you may have!

< 268'm >

Joel Mathew Hegberg
932 N. 12th Street
Dekalb, IL 60115

Delphi : JOELHEGBERG
GEnie : j.hegberg
Internet : JoelHegberg@delphi.com

DON'T LET YOUR SUBSCRIPTION EXPIRE!

The last issue on your current subscription is on your mailing label after your name. If "6/94" is after your name, then the 15 June 1994 issue will be your last.

Micro News

(continued from page 28)

net number, the 'z' is the zone number. If there is a point, it would be "firstname_lastname @ p#. f 834. n102. z1. fidonet.org", just replace the '#' with the point number. The underscore between the firstname and lastname can be replaced by a period, if you like it better.

To go the other way, you need to find a local gateway that has an Internet connection. For example, our local gateway is mcws.fidonet.org, also f 851.n102.z1.fidonet.org (1:102/851). To send Internet/UUCP email, we would send Fidonet netmail to a user named "uucp" at 1:102/851, the first line of the message would contain "To: " with an email address appended, the next line would be blank, then the message would follow on the third line. To send the same message to more than one person, just put another "To: " address underneath the first, then as many more as desired, then a blank line, then the message.

< 268'm >

Programming in "C"

Let's try IF FOR a WHILE

P.J. Ponzo

```
main() {            /* what's in a name ?   */
  int i, j;         /* a bunch of integers. */
 char name[10];  /* an array of 10 chars. */
 for (i=0; i<25; i++) /*print '\n', 25 times, */
    printf("\n"); /* to clear the screen!  */
  printf("\n Type your name : ");
                    /* ask for a name  */
  scanf("%s",&name); /*input the name*/
  for (i=0, j=0; name[i] != '\0'; i++)  {
                    /* a nice for-loop */
    if ( name[i] == 'e' ) /*check for an'e'*/
       j++;         /* if so, increment j */
  }                 /* end of for-loop  */
  printf("\n The letter e occurs %d times in
%s",j,name);
}                   /* end of main() */
```

Let's look at this program. It's supposed to ask for a name, then print out the number of times the letter e occurs in the name.

```
main() {            /* what's in a name ? */
  int i, j, num;    /* a bunch of integers */
 char name[10];  /* an array of 10 chars. */
```

This part is familiar. Note that we allow for a name of 9 characters since the 10th will be the terminating '\0' (remember?).

```
main() {            /* what's in a name ? */
  int i, j;         /* a bunch of integers */
 char name[10];   /* an array of 10 chars.*/
 for (i=0; i<25; i++) /* print '\n', 25 times*/
    printf("\n");  /* to clear the screen!*/
```

We use a for loop to print 25 new lines (thereby clearing the screen..not very elegant, but..). The new thing here is i++ which, in C, means increment i.

```
 for (i=0; i<25; i++) /* print '\n', 25 times*/
    printf("\n");   /* to clear the screen!*/
  printf("\n Type your name : ");
                    /* ask for a name */
```

We now ask for your name by printing:

Type your name :

```
  printf("\n Type your name : ");
                    /* ask for a name */
 scanf("%s",&name);  /*input the name*/
```

Then we wait for the user to type in his/her name (ending with the Enter key), and put this string at memory address &name .

```
  scanf("%s",&name);  /*input the name*/
  for (i=0, j=0; name[i] != '\0'; i++)  {
                    /* a nice for-loop */
```

Now we go through the name[] array, one character at-a-time, starting with i=0 (the first character is the zeroth!). We will count the number of times the letter e occurs and store this count in the variable j, so we also initialize j=0, too! (Note the use of the COMMA between i=0 and j=0).

```
  for (i=0, j=0; name[i] != '\0'; i++)  {
                    /* a nice for-loop */
```

This for-loop will continue so long as the ith character in the name[ ] array is not the NULL '\0'. (Note the construction != which, in C, means NOT EQUAL).

```
  for (i=0, j=0; name[i] != '\0'; i++)  {
                    /* a nice for-loop  */
```

Of course, each time we advance through the characters in the array name[ ] we must increment i (until we reach the end).

```
  for (i=0, j=0; name[i] != '\0'; i++)
                      b   /* a nice for-loop */
```

...our openers for the for...

```
    if ( name[i] == 'e' )  /*check for an 'e'*/
```

And now we check the *i*th character, name[i], to see if its EQUAL to the letter e. (Note the curious way we check for ==). Had we used name[i]='e' it would compile OK, but this actually assigns to name[i] the character 'e' rather than checking for equality! ...and name would be ALL es).

```
    if ( name[i] == 'e' ) /* check for an 'e'*/
       j++;             /* if so, increment j */
```

Now, if we find an 'e', we increment j.

```
       j++;             /* if so, increment j  */
  }                     /* end of for-loop */
```

...and this } ends the for-loop.

```
  for (i=0, j=0; name[i] != '\0'; i++)  {
                        /* a nice for-loop */
  }                     /* end of for-loop */
  printf("\n The letter e occurs %d times in
%s",j,name);
```

After leaving the for-loop, we print the results:

The letter e occurs (j) times in (name)

```
main() {            /* what's in a name ? */
  int i, j;         /* a bunch of integers */
 char name[10];  /* an array of 10 chars. */
 for (i=0; i<25; i++) /* print '\n', 25 times */
    printf("\n");   /* to clear the screen!*/
  printf("\n Type your name : ");
                    /* ask for a name */
  scanf("%s",&name); /*input the name*/
  for (i=0, j=0; name[i] != '\0'; i++)  {
                    /* a nice for-loop */
    if ( name[i] == 'e' ) /*check for an'e'*/
       j++;         /* if so, increment j */
  }                 /* end of for-loop */
  printf("\n The letter e occurs %d times in
%s",j,name);
}                   /* end of main() */
```

We compile, link, execute; and we get:

Type your name : ***Peter***

we type this, then press the Enter key, and get:

The letter e occurs 2 times In Peter

IF (this) do that ELSE IF (this) do that ELSE do that

We may wish to check for several characters (not just 'e'), so we could say:

```
1     if ( name[i] == 'e' )     j++;
                  /* increment j if an 'e' */
2     else if ( name[i] == 'f' )  k++;
                  /* increment k if an 'f' */
3     else if ( name[i] == 'g' )  l++;
                  /* increment l if an 'g' */
4     else ;
                  /* else do nothing */
```

In Line 1 we increment the variable j (which counts the number of times an 'e' occurs. *In Line 2* we increment the variable k (which counts the number of times an 'f' occurs. *In Line 3* we increment the variable l (which counts the number of times an 'g' occurs. *In Line 4* (which is reached only if the character is none of the above) we do nothing. We could have done something interesting, but we should, just once, demonstrate a DO NOTHING statement .... just the *SEMI-COLON!* We would, of course, have declared k and l as int data types.

More stuff like l++

Although we could have incremented i by using i=i+1, we used the increment operator ++. There is also (what else?) a decrement operator —. In fact these can be either pre- or post-operative.

j=i— will assign to j the value of i, then will decrement i.
j=—i will first decrement i, then assign to j the decremented value of i.

Note the convenience of typing antidisestablishmentarianism++ and not antidisestablishmentarianism=antidisestablishmentarianism + 1. Also, the following assignment operators may be used:

x+=5 instead of x=x+5
x-=5 instead of x=x-5
x*=5 instead of x=x*5
x/=5 instead of x=x/5

Tests for EQUALITY && INEQUALITY, etc.

To test for equality of, say, x and 5 we ask if x==5. Had we used something like: if(x=5) then x would be assigned the value 5 (and, of course, x would now BE equal to 5 and the if-statements would certainly be executed). We also use if (x!=5) (for *not equal*) and if (x>5) and if (x<5). We also have:

if (x>5 && x!=7) where && means AND ...so this reads:

if (x is *greater than* 5) *AND* (x is *not equal to* 7)

We also have: if (x=5//x>=7) where// means OR ...so this reads:

if (x is *equal to* 5) *OR* (x is *greater or equal to* 7)

We also have: if (x=x==5) !!!

&& that's all folks! au revoir!

P.J.Ponzo
Dept. of Applied Math
Univ. of Waterloo
Ontario N2L 3G1
CANADA

< 268'm >

Tandy Hi-Res Interface Mod

By-pass Hi-Res With a Switch!

Neal K. Steward

Switching between hi and lores applications requires unplugging and plugging in of the Tandy Hires Interface and your mouse/joystick. To make things easier, a bypass switch can be installed to switch resolution between 640x640 or 64x64 by those with some minor soldering skills. The following article describes this procedure.

DISCLAIMER: Opening the Tandy Hires Interface voids warranty. Also, the author nor publisher is responsible for any damage to the interface or your computer system, so do it right the first time.

TOOLS:
Low wattage soldering iron
Small phillips screwdriver
1/4" drill bit and drill

MATERIALS:
Tandy Hires Interface
DPDT mini switch (RS Part #275-626 or equivalent)
6 short pieces of hookup wire
shrink tubing or electrical tape
Thin PC board grade solder

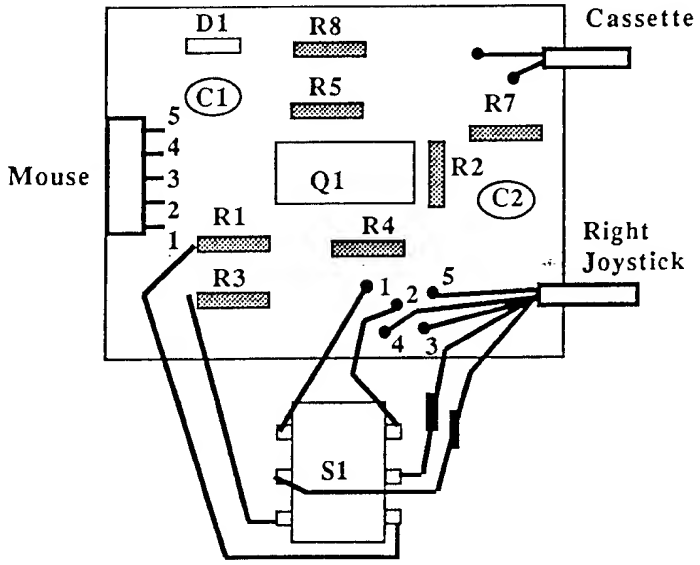
Begin by removing the 4 phillips head screws on bottom of the interface. Carefully pull the circuit board out. Looking at the component side with the 2 short cables pointing to your right, locate the red and blue wires attached to points on the circuit board labeled 1 and 2. De-solder these wires and solder to them two 1.5" pieces of hookup wire, extending their length to about 2". Insulate the solder joints with shrink tubing or electrical tape. Now solder the ends to the common (middle) contacts of a DPDT switch, one per side.

Solder a 2" piece of hookup wire to the "On" end of the switch on the same side as the red wire. Solder the other end to the solder point labeled "1" on the circuit board. Repeat the procedure by soldering another 2" piece of hookup wire to the "On" end of the switch on the same side as the blue wire. Solder the other end of that wire to the point labeled "2".

Locate Resistor R3 on the board. Solder a 2" piece of hookup wire onto the resistors lead closest to pin 1 of the 5 pin socket. Solder the other end to the side of the switch that the red wire is on. Just above R3, solder another 2" piece of hookup wire onto the lead of R1 closest to pin 2 of the socket. Solder the other end of that wire to the side of the switch that the blue wire is on.

Check your wiring carefully, then connect as usual and test. If it works with both hi and lores applications, then proceed. Drill a 1/4" hole on one of the long sides, 3/8" from the top and centered side to side to allow room for the switch and the circuit board. Place switch in hole from inside case and tighten nut. Carefully put circuit board and wires in top part of case, aligning the short cables with notches. Replace bottom and 4 screws.

< 268'm >



Basic09 In Easy Steps

Chris Dekker

Differences between Disk Extended Color Basic, IBM Basic, and Basic09

Now that you know a little about Basic09, it's time to get Basic09 up and running. We will be making a bootable diskette (so you can start it by simply typing DOS) that holds Basic09 and it's supporting programs. Also the computer will boot into OS-9's windows instead of the 32 column VDG screen.

All of this is nowhere near as difficult as it sounds. What you need is your OS-9 system master disk, your Basic09/config disk and a blank diskette. The usual process would be to run the config utility included in OS-9, but I find that cumbersome. Instead, I will give you a series of commands that will alter a number of device descriptors. This is the equivalent of a series of pokes under DECB. The reason we can't use pokes with OS-9 is that it's code is position independent. This means that, except for OS-9, no one has a clue where to find the actual code. OS-9, however, keeps the starting address of every module in memory neatly in one big table. Since all device descriptors have a predefined and strictly adhered to format, we can find the correct byte by specifying an offset from the descriptor's starting address.

Well,, enough theory for now. Boot up your computer from your OS-9 system master disk and enter the date and time when you are prompted for it. Then at the OS9: prompt type:

modpatch <ENTER>.

Modpatch is an OS-9 utility that will do the dirty work for us (as you probably guessed there is more to the job than I described above). At the end of all of the following commands you must press the <ENTER> key, even if it isn't specified. Modpatch has a strange way of echoing characters but just type the following lines and keep on going (Please DON'T type the comments too!!):

```
l d0      (l as in link)
c 14 00 03  (6 ms step rate)
c 18 23 28  (40 track drive)
c 19 01 02  (double sided)
c 1A 00 01  (verify off)
v         (verify module)
```

If your drives are really old (SS 35 track, 30 ms step) don't make the above changes. Verify OFF will greatly speed up OS-9's drive access since it no longer has to reread every sector written to disk. Most newer drives are reliable enough anyway. I have run my drives like this for a few years now and never experienced problems because of it (I will also keep my fingers crossed from now on). Make sure you don't forget to type the v command. This verifies the updated module. If you forget this you won't be able to boot your computer from your Basic09 disk.

Since OS-9 has two more descriptors for floppy drives in memory you must update those also. Type the same series of commands but change the first line to l d1 and l dd to access those descriptors.

Next comes your printer. This is locked in at 600 baud but most printers can take their data in a lot faster.

```
l p       (link p)
c 27 02 xx (changes baud rate)
v
```

Instead of xx you must enter a code number that represents the printer's baud rate. A table of values can be found on page 6-96 (or 6-103; see **table 1**) in the OS-9 commands section of your manual.

Last but not least the terminal descriptor:

```
l term
c 19 01 00  (disable end-of-page pause)
c 26 01 80  (boot into windows)
c 2C 32 28  (40 column screen)
c 33 07 00  (foreground: white)
c 34 04 02  (background: black)
c 35 04 02  (border: black)
v
<BREAK>     (press break key to exit)
```

Note that the numbers dealing with the colors are pointers to register numbers and NOT the color codes themselves. If you want your computer to boot up with a 80 column screen, replace 28 in that line with 50.

Now that you have adapted the crucial parts of your system it is time to save those changes to disk. First we must format a blank diskette. The following commands assume you have 2 disk drives. If not you will have to replace d1 with d0 and swap disks if necessary.

Type: format /d1 "Basic09"

Answer the prompts of the format utility. Once your disk is formatted you type:

```
cobbler /d1
makdir /d1/CMDS
CHD /d1/cmds
copy /d0/cmds/shell shell
copy /d0/cmds/grfdrv grfdrv
```

At this point your diskette is bootable. You may want to put it in drive d0 and press the reset button. The computer should boot without any problems. You can use this procedure also for creating other disks since all bootable disks must carry the same information.

Note that the format of your new disk is DS 40 track. Now that your disk is ready you must copy Basic09 and some other programs to it. Start by putting your Basic09/config disk in drive d0 and your new disk in drive d1. Once again type:

```
CHD /d1/cmds
setime  (answer it's prompt)
```

Then, using the same type of copy commands as above, copy the following programs: Basic09, Runb, gfx, gfx2, inkey and syscall. To conserve memory we will merge some utilities. Type: merge gfx gfx2 inkey syscall > utilities

Now put your OS-9 system master disk in drive d0 and type:

```
CHX /d0/cmds
attr utilities e pe
makdir /d1/SYS
CHD /d1/sys
copy /d0/sys/stdfonts stdfonts
copy /d0/sys/stdptrs stdptrs
```

You can copy the stdpats_2, stdpats_4 and stdpats_16 files in the same way if you want. These files contain fonts, cursors and fill patterns to be used on graphics screens. To keep things tidy on your new disk you will want to create a directory to hold your programs: makdir /d1/SOURCE

The last thing we have to do is automate some tasks; like loading the various programs on startup. For that purpose we create a file called startup in the root directory of the Basic09 disk. Type:

```
CHD /d1
edit startup
```

At the E: prompt type the following lines (start them with a space):

```
loadutilities
setime </1
load basic09
```

Exit this program by typing q as the first character of the line and then press <ENTER> Your Basic09 system disk is now ready to go. Next time you want to use it, you can boot your computer from this disk, enter date and time and then type the following commands:

```
CHD /d0/source
basic09 [#16k]
```

Note that the memory modifier is optional. By default Basic09 starts with 8k of space available. With the modifier you can give it up to 40k of space. At least if you don't want to use syscall or the gfx modules. For most purposes a value of 16k or 24k is more practical.

Basic09 consists of two parts: the editor/compiler and the code that actually runs the I-code. This latter part is also available in a separate module called Runb. This module is used to run packed I-code.

That's it Folks!! Next time we'll try to run some code.

< 268'm >

Table 1:
OS-9 Printer Baud Rate Values:
0= 110 2=600 4=2400 6=9600
1=300 3=1200 5=4800

Support for OS-9 Level II: Shell+

Robert Gault

Using "shellsubs" To Create New Windows

Last issue we saw how to obtain the number of the next available window under OS-9 Level II. Now I'll show you how to use the info to create windows with unique border colors.

Before Shell+ the only approach was to hand enter the Display or Wcreate commands. Now we can have a shellscript which includes one of the two following lines:

1) iniz w%0; display 1b 20 20 05 01 82 07 1b 31 7 %1>/w%0

2) iniz w%%0; display 1b 20 20 05 01 82 0 7 1b 31 7 %%1>/w%%0

Line 1) uses the Shell+ user variables but as these can not easily be auto loaded, this approach is of limited value. Line 2) uses the Shell+ shell variables. Since these can be automated it is just the method we need if we can find a way to load the shell variables.

In the listing section are two Shell+ subroutines designed to do the job. Both take input from the standard input path and place data in the correct shell variable. ShellAlph expects alphanumeric data and sends it unchanged to the shell variables. ShellHex expects decimal numbers, removes leading zeros, and sends the hexadecimal equivalent to a shell variable. The syntax follows:

```
          shellalph3<enter>
OS-9 prints new line and waits for data
          13<enter>
var.? will show: var.3=13
          shellalph3<enter>
          <enter>
var.? will show: var.3=
shellalph 0 3 1 9<enter> or 0319<enter>
          123<enter>
          456<enter>
          <enter>
          789<enter>
var.? will show: var.0=123
           var.1=
           var.3=456
           var.9=789
```

What is important about both shellsubs is that keyboard entry is not necessary. A shellscript can contain the following:

getnw ! shellalph 0

Below is the Shell+ shellscript I use to get text windows with different border colors. Each line is commented:

```
* start an 80 column window in the next
* available space with new border color
* "prompt" comes with Shell+; *"echo"
* without a CR
* var.3 gets count from keyboard
prompt How many windows should I create?
;var.3
* next line is Shell+ goto label
*main
* next lines convert to Base 0 math
if+%3<>00000
dec.3
* pipe output from getnw (get next window)
* to shell variable %%0
getnw ! shellalph 0
* next line transfers shell variable %%0 into
* user variables %0 and %1
var.0=%%0; var.1=%%0
* next line is goto label; loop assigns colors to
* each window
loop
* Update window value counter
if+%0<>00000
dec.0
* next line is ethetic choice; use every other
* color; palette starts at w#
inc.1
inc.1
goto loop
endif
* next line takes %1; removes leading 0s;
*converts to hexadecimal because Display
*requires hexadecimal for mat. NO SPACES
*  between %1  and  !
echo %1! shellhex 1
* initialize & start 80by24 text window;
* border = slot7; slot7 palette = %%1
iniz w%%0; display 1b 20 20 05 01 82 07 1b
31 7 %%1>/w%%0
* I like different action keys than stock OS-9
xmode /w%%0 reprint=9 dup=19 pause
* tell user what just happened
echo 80col. text/w%%0 started with process
* start immortal shell in new window and
* supply process number for above echo
shell i=/w%%0&
* process next window if >1 requested
goto main
```

Listing 1

```
          nam   shellalph
          ttl   generic shell subroutine - by Robert
Gault  Sept.1993

* Get input from standard path and send it to
* Shell+ as shell variable.
* Correct sv# is determined; up to 81 bytes
*data processed per var.;
* multiple variables handled.
* Input can be from pipe or keyboard. Each
*data block must terminate with
* a <CR>.
*
* Usage:  ex.  shellalph 0 3 1 9
*  shell expects v 0,3,1 &9  to be entered from
* keyboard.
*          user types:   123<CR>
*                        456<CR>
*                        <CR>
*                        789<CR>
* var.?     will show shell variables:  var.0=123
*                                       var.1=
*                                       var.3=456
*                                       var.9=789
*
*   ex.  dir ! shellalph 0 1 2 3 4 5 6 7 8 9
*  shell gets v0-9 from pipe
* var.?     will show first 10 lines of the directory
* This example works but is not recommended
* for if DIR requires  more than 10 lines, it can't
*  send the entire report so DIR gets hung up
*  until user types <CR>.
*
* NOTE: It is not necessary to seperate variable
*  numbers nor have them  in order.
*  ex. shellalph 1 2 3      and      shellalph 312
*  fill same var.s  but order of fill is different.
*********************************************************
*
* Entry :
*         X - parameter start
*         D - parameter size
*         U - shellsub buffer start
*         Y - subroutine start address

        ifp1
          use   /dd/defs/os9defs
        endc

ShellSub set    $50
tylg      set   ShellSub+Objct
atrv      set   ReEnt+rev
rev       set   2

vlen    equ   81       maximum length of a shell
variable

* No data field for subroutines.
          mod    eom,name,tylg,atrv,start,0

name      fcs   /ShellAlph/
          fcb   1

start     equ   *
          cmpd  #0       quit if no parameters
          beq   quit
          tsta
          bne   quit     quit if too much parameter
info; >255
          pshs  x        make room on stack
mainlp    lda   ,x+      read param. character
          stx   ,s       save updated pointer
          cmpa #$20      skip spaces
          beq   mainlp
          cmpa #$0d      eof
          beq   exit
* ASCII # into binary;
* assumes single digit numbers
          suba #'0
          bcs   error    not a number
          cmpa #9
          bhi   error    not a decimal number
* index correct variable
          ldb   #vlen    space reserved for a shell
variable
          mul            calculate an offset
          leax  d,u      point to correct variable
```

```
      lda #0      get values from standard input
or pipe
      ldy #vlen     max length to read
      os9 I$ReadLn  read line through CR or
vlength
      bcs error
      ldx ,s        recover parameter pointer
      bra mainlp    check for more variables

exit    equ *
error   puls x        reset stack; both an error
and normal exit
quit    rts           return to shell+; short path
exit
      emod
eom     equ *
```

Listing 2

```
      nam   shellhex
      ttl   convert input to hex value in shell
variable; R.Gault 1993

* Get input from standard path and send it to
*  shell as shell variable. Correct sv# is deter-
*mined; up to 5 bytes dec.# processed per
* var.; multiple variables handled. Input can be
* from pipe or keyboard. Each data block must
* terminate with a <CR>.
*
* This subroutine is particularly useful with
*commands like Display when using calcu-
* lated shell variables.
* ex. var.0=10    decimal
*    inc.0
*    var.0 now =00011 but this number can not
* be used with Display because it is decimal
*  and has leading zeros.
*
* Usage: ex. use a routine to get next available
* window # into variable %%0;  make border
* color = window#+2
*
*   *  %%0 contains next available window
*      var.0=%%0; var.1=%%0
*     *loop
*     if +%0<>00000
*     dec.0
*     inc.1
*     inc.1
*     goto loop
*     endif
*   *  next line MUST NOT have any spaces
* between %n and pipe symbol
*      echo %1! shellhex 1
*  *  next line MUST do an (iniz; 1b 20) before
* the 1b 31
*      iniz w%%0; display 1b 20 2 0 0 50 18 2
* 0 7 1b 31 7 %%1>/w%%0
*     shell i=/w%%0&
*
*
* Entry :
*        X - parameter start
*        D - parameter size
*        U - shellsub buffer start
*        Y - subroutine start address

      ifp1
        use   /dd/defs/os9defs
       endc

ShellSub set   $50
tylg    set   ShellSub+Objct
atrv    set   ReEnt+rev
rev     set   1

vlen    equ   81      maximum length of a shell
variable

* No data field for subroutines.
        mod   eom,name,tylg,atrv,start,0

name    fcs   /ShellHex/
        fcb   1

exit    equ *
error   leas 2,s      reset stack; both an error
and normal exit
quit    rts           return to shell+; short path
exit

start   equ *
        cmpd #0       quit if no parameters
        beq  quit
        tsta
        bne  quit     quit if too much parameter
info; >255
        pshs x        make room on stack
mainlp  lda  ,x+      read param. character
        stx  ,s       save updated pointer
        cmpa #$20     skip spaces
        beq  mainlp
        cmpa #$0d     eof
        beq  exit
* ASCII # into binary; assumes single digit
*  numbers
        suba #'0
        bcs  error    not a number
        cmpa #9
        bhi  error    not a decimal number
* index correct variable
        ldb  #vlen    space reserved for a shell
variable
        mul           calculate an offset
        leax d,u      point to correct variable
      lda #0      get values from standard input
or pipe
        ldy  #6       max length to read; 5+CR
        os9  I$ReadLn read line through CR or
vlength
        bcs  error
* shell v. now contains ascii decimal number;
*  convert to ascii hex via binary
        clra
        clrb
        pshs d,x      make room on stack
cnv     lda  ,x+
        cmpa #$0d
        beq  endcnv
        suba #'0
        bcs  badnum
        cmpa #9
        bhi  badnum
* this next section does a (16-bit value in
*  reg.S)*10+reg.A
        pshs a
        ldd  1,s
        lslb
        rola
        lslb
        rola
        lslb
        rola          ans=8d
        addd 1,s
        addd 1,s      ans=8d+2d=10d
        addb ,s+
        adca #0       ans=10d+a
        std  ,s
        bra  cnv      loop back for more digits

endcnv  puls d        reg.D=binary number
* now convert binary number to ascii hex
        ldx  ,s       get copy of pointer to shell
variable
        cmpd #0       trap special case #=0
        beq  single0
        pshs b        save LSB
        bsr  hex      convert reg.A to ASCII hex
in reg.D
        std  ,x++     save MSB and update
pointer
        puls a        recover LSB and process
        bsr  hex
      std  ,x     shell variable now contains raw
ascii hex#
* now remove leading zeros if present by
successive left shifts
        ldb  #3       worst case 0001
        pshs b
shift1  ldx  1,s      point to start of shell variable
        ldb  ,s       init. counter
        lda  ,x       test leftmost digit
        cmpa #'0      leading 0?
        bne  endshft
shift2  lda  1,x      move digits one place left
        sta  ,x+
      decb        until all remaining digits shifted
        bne  shift2
        dec  ,s       repeat until 3 block shifts
made or leftmost # <> 0
        bne  shift1
* shell v. now has normalized ascii hex#; insert
*  required terminal <CR>
endshft puls b,x
        incb
        abx           reg.X now points past ASCII
number
nextpar lda  #$0d     set the required <CR>
        sta  ,x
        ldx  ,s       recover parameter pointer
        lbra mainlp   check for more variables

single0 lda  #'0      special case of 0
        puls x
        sta  ,x+
        bra  nextpar

badnum  puls d,x      not a decimal number
        bra  nextpar

* convert binary number in reg.A to ASCII
*  hexidecimal in reg.D
hex     tfr  a,b
        lsra          get MSN
        lsra
        lsra
        lsra
      bsr  hcnv     convert to hex and swap regs
A&B
        anda #$f      get LSN
hcnv    cmpa #9       convert it
        bls  notletr
        adda #7
notletr adda #'0
        exg  a,b      second pass gets number
realigned
        rts
        emod
eom     equ  *                          < 268'm >
```

Reviews

QuickLetter - Basic Word Processing for OS-9 Level II

F.G. Swygert

Quick Letter was designed for all those light word processing jobs that don't really require a full featured word processor. If you need "to write a book or thesis" get a full featured word processing package. "If you want to write a letter in the next 20 minutes... use QuickLetter" (quotes from manual cover).

A CoCo 3 with at least one disk drive and an 80 column monitor (composite or RGB) is required. The program also requires access to *Runb*, *syscall*, and *gfx2* (all on the Basic09 disk). These and any other modules in memory must take up no more than 24K of system memory, as QL itself takes up 40K. The three Basic09 Modules can be merged into one file easily with the MERGE command (page 6-68 of Commands Reference, OS-9 Manual; page 8 of FARNA Systems QRG).

Although not recommended for heavy duty word processing, QL has an 18000 character buffer. The default settings allow roughly 1500 characters per page, so that means about 12 single spaced pages can be typed into a single file before having to save. This is about the same as Telewriter-64 under DECB on a 64K CoCo2. I have used TW-64 to write a book, so it can be done.

After loading the required Basic09 modules, insert the QL disk, change the execution directory to QL's CMDS directory ,and type QL. The program is hard coded to expect a data disk in your /dd drive (drive /d0 for most of us, possibly a hard drive though). The data disk must have a directory by the name of "textdata" to save files to, so make sure you create this directory before trying to save any files.

The first screen that appears offers four choices: Save (CTRL S), change system settings (F1), main menu (F2), or exit (CTRL F2). TEXT ENTRY appears on screen in all capital letters. This indicates that QL is ready to receive text now. All one needs to do is start typing.

Let's see what the "change system setting" selection does. Pressing F1 opens a window in the center of the screen. The window contains the default file settings for margin, page length/width, line spacing (in 72nds of an inch), and top and bottom margins. The default settings display 72 characters per line on screen. Word-wrap and hyphenation are not automatic, but the program activates a "bell" for the last five characters on a line. So the end of a line won't sneak up on you and hyphenation and/or word-wrap can be anticipated by the writer. The bottom two lines of the screen always display the number of characters, lines, and pages entered; current page length (in lines), and how many lines to the end of the current page.

Only three line widths are supported- 80, 96, and 136 characters. These widths were chosen to support the three print densities supported by almost all 80 column printers: standarad (10 cpi), elite (12 cpi), and condensed (17 cpi). Use the arrow keys to highlight the item to hange the n press C and enter the new value. F1 takes us back to text entry mode.

Striking F2 for the main menu opens a nother center screen window. This menu has seven number selections: load, save, edit, append, overwrite, acces OS-9 shell, and exit. Press the number desired (no need to press enter). Again, make sure there is a "textdata" directory on your default drive before operating on any file!

Once data is typed in, the file must be saved before any changes can be made- you can't simply back the cursor to previously typed text. Instead, you must go to the main menu and enter the "edit" mode. Once in edit, the cursor is postioned to one character before the text to be edited. Then the "@" key is pressed and a window is opened with another menu. The edit options are insert, delete, change, update file, delete line, mark end of text, and move block. SHIFT @ will return you to the edit mode. To continue with the current file, exit edit mode (SHIFT @), go to the main menu (F2), and choose append text (4). This takes you to the end of the current text and into the text entry mode.

Option 7 (move block) opens another menu. The selections are start of block, end of block, insert block, cut block, copy block, delete block, and exit. Again, sinply press the number of the selection desired. Position the cursor where a block is to be started while in the edit mode. Go to the edit menu (@) and select move block (7) then start of block (1). Move the cursor to the end of the desired text, and repeat the process to mark the end of a block. Repeat again to cut/copy/delete the marked block of text. Sounds involved, but is really pretty simple after a couple tries.

Printing a file is not handled by the main text editor or a text formatter. All the formatting information is embedded in the data file. Printing is handled by a utility called QLPrint. The file to print can be entered at the command line (qlprint file1) or the program will ask for a file name if one is not entered at the command line.

QL supports any printer. QLPrint reads a table of printer codes that must be stored in the textdata directory with the name "printcodes". Files for three printers are included on the distribution disk: DMP105, NX2420, and proX24E. The NX2420 file emulates an Epson LQ850/860 and should work with most printers supporting Epson codes. The proX24E file supports most printers that have an IBM emulation mode. Simply copy the file you wish to use to /dd/textdata/printcodes.

If none of the included files seem to support your printer, the "printersetup" utility allows a new file to be created. When printersetup is run, it will allow editing of an existing printcodes file or create a new file if none is found. Get your printer manual out and type in a list of commands as prompted and you will have a custom driver that will take full advantage of your printer. An 80 column screen is required by this program. If it is started in anything else, it will create a new window, so if the screen goes blank press CLEAR to change windows.

Utilities are included to add additional functions to QL:

Addpage will add page numbers to a file. You will be prompted for the starting page number and file name. Page number data will be added to the specified file.

Strip converts a QL text file to a standard ASCII file. The file header (which containd formatting info), embedded printcodes, and left margin codes are removed.

Erase overwrites the specified file with logical zeroes and then calls the OS-9 del command to delete the file.

Qlist works similar to the OS-9 list command. It simply lists a QL text file to the screen or printer. A file sent to the printer will be unformatted, as Qlist ignores all formatting codes.

Setlm allows a quick change of the left margin setting.

Envwriter prints addresses directly onto an envelope. The return address must be in an ASCII file called "returnadress" in the textdata directory (create with OS-9's build or QL and strip). The main address can be typed in or pulled from a letter.

Qlmerge merges two QL files into one. It is used instead of OS-9's merge to handle the file header and embedded codes. Only two files can be merged at a time.

Follow a few guidelines, and QL becomes a very easy to use word processing tool. I suggest one type in a rough draft and finish a file before editing. If you make a mistake, simply skip a space or line and continue typing. When finished, save the file then go into the edit mode to correct the mistake(s). Who said word processing under OS-9 had to be difficult?

C. Dekker $19.95
RR #4
Centreville, NB E0J 1H0
CANADA

< 268'm >

The MM/1 Column

The MM/1 is delayed by the original board manufacturer.

In January of 1994 , BlackHawk began offering the MM/1 for sale once more. I did so following statements by a representative of Incor Systems Corporation (the board manufacturer) that they had 50 complete MM/1 systems ready to ship. They requested only that I pay them cash in advance. I have received a nice response to this offer - several system sales and 2 I/O card sales. So, in early January, I sent an order for the first I/O board sale to Incor, accompanied by a cashiers check.

In late January, I received a shipment from Incor. Delivery took 20 days, and when I opened the package, it was not the item I ordered. When I called to find out why, I received a very unpleasant surprise! Representatives of Incor told me that they did not have ANY completed I/O boards in stock. In addition, they had no bus cards at all! It took a lot of phone calls and several days to find out that Incor now claims to have 42 MM/1 CPU boards in stock, and 63 bare I/O boards in their possession. Moreover, in spite of the company sales reps word, Incor would not guarantee that they would assemble the I/O boards they did have! If I wanted to have them populate those boards, I would have to fax them copies of my orders, so they could verify I had sales.

OK, with IMS's past record, this was an understandable and reasonable request.. The next Monday, I faxed my orders to them. Over the next week I waited the only call coming from Donald Adams, who called to ask why Incor was calling him to ask if he sent them a fax?? Well, on the 9th of February, Incor called to say they would not fill my orders.

So, where does BlackHawk Enterprises and the MM/1 go from here? Onward and upward! I have made arrangements to start production on MiniBus cards. The unfinished I/O boards at Incor are property of BlackHawk Enterprises, Inc. per our deal with Interactive Media Systems, Inc. Incor's plant manager has assured me that any items in their possession which were formerly IMS property would be shipped to me.

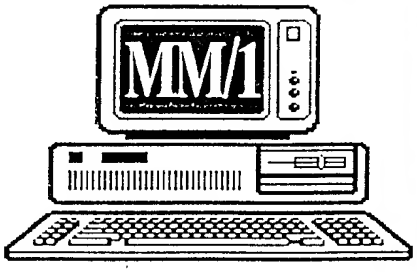
So, at this time, I am looking for a cost effective approach to continue assembly of MM/1 Systems. CPU boards will continue to come from Incor for the time being. I've made arrangements to produce 8 megabyte bus cards. We can assemble the bus cards here at our own facilities. Arrangements for the I/O boards are being considered. Our sales offer for systems still stand, and orders received will be used to help finance these efforts, as I noted in my advertisement, and in my previous special report to 68 Micro's.

It is possible at this time to fill orders for Personal MM/1's from Incor with little delay (up to 6 weeks time). However, orders for Extended Systems will be delayed until I can work around the latest snags with the I/O boards, and make whatever alternative production arrangements are necessary. All arrangements will be consistent with my original intent of reducing shipping time to 2 weeks by accumulating a local inventory as soon as possible. I hope that, with the continued support of the OS-9 community, the MM/1 can be returned to full production by June or July of this year. Thank you all.

Editor's note: *I recently talked with David about this situation. He is extremely displeased with Incor at the moment (frustrated is more like it, and even that is an understatement!), and understandably so. When David first approached Incor , before he had committed to the MM/1 at all, they were agreeable to selling all stock and not holding him liable for the IMS debt load. They apparently changed their minds after selling the system boards. Incor would now like BlackHawk to assume IMS's old debt before delivering any of the I/O boards. They are still willing to sell the populated system boards though.*

David's guess is that Incor will probably change their minds again, but for now expect to sell the system boards and create a pent-up demand for the I/O expansion boards. Luckily, enough brand new I/O boards have been located to fill the orders generated by recent announcements, and profits from those sales are being put into a new source for the boards, by-passing Incor altogether, unless they reconsider soon. If this happens, count on a few changes being made to the boards- like all being capable of 8MB of RAM.

< 268'm >



Micro News

The latest news on the MPC601 is that in real life operation the Intel Pentium is roughly half as fast as a PowerPC based Mac in integer performance and three to four times slower in floating point operations. These times came from Fractal Design Corp. using ports of their Painter software. Even when the Painter Windows code was recompiled specifically for the Pentium speed improved only by about 10%. The comparisons are between a 60MHz Pentium and PC601.

The PowerPC series is competing with the Intel Pentium in price as well as performance. Currently, a 66MHz Pentium chip sells for $871 whereas IBM is shipping 66MHz PC601s at only $374 (both in OEM quantities). A typical Pentium system now sells for $3500-$4000, but the PowerPC Mac introduced this month sells for closer to $3000 AND has a performance advantage. Just like "our" Motorola to deliver the most "bang for the buck"!

What is Intel doing about this? Well, I've already reported that the PC601 is being boosted to 80MHz. So Intel is doing the same to the Pentium- boosting it to 100MHz in the second quarter of this year (the P54C), about the same time the 80MHz PC601 will debut (the 80MHz PC601 is comparable in performance to the 100MHz P54C). Intel is also making a 486DX4/100MHz, which will push the 486 series into the current Pentium performance range (60MHz P54), and bring the pricing of high end 486 systems in the $2,000 range, with Pentiums starting at only $500 more. They are also introducing 486SX2 processor in 50 and 60 MHz models (available this month). Intel is known to have lots of room in their high end profit margins to allow for lots of cost cutting also, and is likely to do so when PowerPC systems start entering the market in large quantities. For the mean time, expect them to hold onto those high profit margins as long as they practically can. There is obviously a lot of strategy involved in the development and pricing of microprocessors!

Apple spotlighted PowerPC systems and upgrades at the MacWorld convention in San Francisco the first week in January. Upgrade boards for current Mac platforms will range from under $700 to around $2,000- enough under the cost of a new PC601 Mac to make the upgrade worthwhile if one really needs/wants a PowerPC platform. Several PowerPC software development tools were also released for developing native PowerPC applications.

There has been an interesting development in the Intel vs. AMD (Advanced Micro Devices) 386 code lawsuit. The circuit court in san Francisco recently lifted a stay on the suit to allow the case to move forward. The stay was originally granted to allow another suit Intel brought against AMD on the 287 (math coprocessor) to be settled. Court was under way on that case when the stay was lifted, and is likely to be finished before the 386 case goes to court. A case against AMD is expected to begin this month (may have begun by the time this is read) on the 486.

Intel is keeping a financial cloud over AMD with these trials. AMD can't get financing to build new plants because damages as high as $2 billion could be awarded to Intel. This keeps AMD from competing as much in the 386/486 market as they would like.

The disputes key in on language in a 1976 cross-licensing agreement between Intel and AMD that was renewed in 1982. The agreement gives AMD rights to microcode used in Intel "microcomputers" and "peripherals". AMDs main argument is that the terminology has changed- in 1976 "microcomputer" was used when referring to what is known as a "microchip" (microprocessor) today.

What Intel is trying to do is keep anyone from competing with them in the PC compatible market. It really amounts to Intel wanting a monopoly on the US computer market. Don't get me wrong, Intel has a right to protect what is actually its property that is copyrighted (the 287/386/486 microcode). But squashing all competition is a little overkill.

The Bell System was broken up because someone decided they were to big and controlled to much of the market. If Intel wins these cases, they are likely to try going after Cyrix, another "compatible" MPU manufacturer. Maybe then someone will decide that maybe Intel controls to much of the MPU market.

AMD has plans to develop their own microprocessor that is Windows compatible in the future. That would eleviate most of the "Intel compatible" problems they have now. I guess Microsoft will try going after them then.

Now here is something I almost missed! CompuServe (a nationwide electronic information system similar to Delphi, only a bit bigger), notorious for high on-line rates, is slashing prices! Hourly connect times have been reduced to the following effective 06 FEB 94:

300-2400b to $4.80 from $6/$8/$8

9600b to $9.60 from $16.00

In an interview published in a late January MacWeek, Bill Gates (that's right.. the father of Microsoft) pledged that there would be a Mac/PowerPC native mode version of Microsoft Office sometime in 1994. When asked about porting the next version of Windows (code named "Chicago") to the PowerPC, he answered that there didn't seem to be any reason to port the system to another processor (other than Intel). Well, let's hope that Motorola, IBM, and Apple can change that view!

Do you REALLY want a MPC601 (PowerPC) computer? One that will run Mac System 7, DOS, and OS-9000? Do you have $2100 available? If so, go to your local Apple dealer and ask for a PowerMac 6100/60. It comes with a 160 MB hard drive and a 60MHz MPC601. Not fast enough? Ask for the PowerMac 7100/66 ($2,999) and get a 66MHz machine and 250 MB hard drive, or the PowerMac 8100/80 ($4,499) with 80 MHz MPC601 and 250 MB hard drive. All come with 8 MB of RAM, keyboard, and VGA monitor. The pricing appears to be very good, say industry sources. These are all rated as mid to high range systems, competeing with fast 486 and Pentium machines. They should be available by the time you read this. The retail prices are good enough that you shouldn't expect street prices to be much lower. That's the way it really should be - sort of like buying a GM Saturn car- priced right the first time! There will also be PowerPC upgrade boards available for the LC 475, 520, 550, and 575; Performa 550; and Quadra 605 in March. There is a rumor going around that anyone who purchased a Quadra (except the 605) between 01 January and 14 March will be getting a $300 rebate on a PC601 board.

I have some good news for C programmers! A source for source code has come to my attention. This isn't free code, but it may be just what you need to get that program you've been working on finished, or be the start of a new program. Some of the available code includes: Minix OS (Unix-like OS; $150), Delorie GCC (for MS-DOS, includes C++, assembler, etc., complete source code- should be adaptable to 68K; $150), CPPCOMM (C++ serial comm library, includes X/Y/Z modem; $75), Backup/Restore (by Blake McBride, multiple volumes, compression, and encryption; $50), COP (poor man's C++, C macro pack implements C++ in C; $35), Spell Pack (6 spelling programs, hyphenator, utils, 60K word list; $30), Bywater BASIC (complete interpreter & interactive programming environment; $20), and a lot more. Much of this is originally coded for MS-DOS, Windows, OS/2, and Unix, but should be modifiable for 68K machines. I had a full page sample listing and was told there was "much more". How much more? Write The Austin Code Works, 11100 Leafwood Lane, Austin, TX 78750-3587 and ask for a free catalog to find out. If you have access to Internet, their e-mail address is info@acw.com.

The following was relayed to me by Jason Reighard in Ohio. It is a message posted by Yoshio Nakamura (the same person who first outlined the expanded Hitachi 6309 information) in Japan stating how to send Internet mail to a person on FIDO net and vice-versa. Thank you Jason and Yoshio!

From Internet/UUCP to Fido, if you have a site like 1:102/834 (Yoshio Nakamura's FIDO address), then the Internet address would be "firstname_lastname @ f834.n102.z1.fidonet.org". The 'f' is the node number, the 'n' is the

(continued on page 20)

If you have new soft or hardware products, let us know! We will gladly print a free blurb for you here in MicroNews whether you advertise or not (though we will be happy to have your ad also).

THE SWAP SHOP

Buy, sell, trade hard/soft ware! Ads are free. **Only those vendors with at least a 1/4 page ad may place classifieds without special permission.** For these, short ads will be free as space permits. PLEASE drop us a post card or E-mail if/when an item sells or is found!

FOR SALE

Multi-Pak, 26-3124 (small), upgraded for CC3- $70; OS-9 Level II- $35; Multi-Vue-$10; Koronis Rift-$10; Rogue-$10. Shipping included to US. John Gar, RR1 Box 141, Newel, SD 57760

Replacement 68B09E for all CoCos. $7 each includes S&H. Timothy D. Boos Sr., Rt. 1 Box 155, Peculiar, MO 64078

Complete CoCo 3 system; CM-8, two drives, 512K, lots of software including OS-9 Level II. Asking $350, will pay shipping via UPS ground. Contact Frank Swygert, 904 2nd Ave., Warner Robins, GA 31098 (912-328-7859)

SALE OR TRADE

CoCo 3, 512K, CM-8 RGB monitor, FD-501 disk drive (2 drives), Ken-Ton SCSI hard drive interface, Y-cable, RGB-DOS in ROM, 40MB hard drive (factory RGB-DOS setup for DECB), lots of software (send SASE for listing). Will take best offer. Alan Clarey, 1012 Borrette Lane, Napa, CA 94558

26-3124 MPI - $45; Tandy RS-232 Pak - $30; 2 Touch Pads - $20 (or trade for pair of deluxe joysticks); V-Term - $15; VIP Writer - $15; Prices include S&H in the US or Canada. George Demers, 603-726-3638

I have approx. 24 never used Tandy CoCo 3 programs for sale or trade for CC3 soft/hardware. Send SASE for list. John R. Mott, Jr., Box 26246, Phoenix, AZ 85068-6246

WANTED

Tandy OS-9 C Compiler with documentation. Contact Joe Charbonneau 527 Jarvis St., Windsor, Ontario N8P 1C8 (Canada); Phone 519-735-8630

Used CoCo hard/software. Will buy by piece or entire collections. Rick Ulland, 449 South 90th St., West Allis, WI 53214

This Issue's "microdisk"

OS-9 Content:

PLAY.BAS	POPUP.C
HTOI.BAS	SHELALPH.BAS
ITOH.BAS	SHELLHEX.BAS
POPUP.BAS	

Disk BASIC Content:

6309INFO.ARC	TC31.BIN
GETERM.ARC	

*Note: TC31.BIN only works with 512K. GETERM. ARC (CoCo 1/2/3 terminal program) was compressed with TC3.BIN, but TC31 will decompress. TC1/2 and TC3 won't decompress TC31 archives.

"microdisk" is available for $40 per year , $21 for six months, or $6 per issue. Overseas must add $10/year, $5/six months, $1/single issue for air mail delivery. "microdisk" is NOT a stand-alone product, but compliments "the world of 68' micros" magazine.

Disk BASIC users, don't criticize the quality of programs and articles... DO SOMETHING ABOUT IT! See 'Submitting Material', this page.

MultiBoot by Terry Todd & Allen Huffman

Now have up to SIXTEEN bootfiles on your startup disk!

Hot off the assemblers and compilers is a great must-have utility which lets you have up to 16 bootfiles on one disk! No more boot disk floppy-swapping! MultiBoot will install itself to a cobbled boot disk and, upon typing "DOS", will greet you with a scrolling menu of available bootfiles!

OS-9 Req: CoCo3, OS-9 Level 2.....$19.95

Towel by Allen C. Huffman

The first EthaWin program - a disk utility for OS-9.

A program no intergalactic hitchhiker should be without! Use a mouse or keyboard hot-keys to perform common file and disk commands from pull-down menus. Tag multiple files for Delete, Copy, Rename, etc., and even have point 'n click disk Backup, Cobbler, Dcheck and other commands. User menu lets you specify up to seven of your own commands to execute. Runs under the EthaWin interface on a high-speed text screen. All commands/colors configurable.

OS-9 Req: CoCo3, OS-9 Level 2.....$19.95
OS/K Req: MM/1 or K-Windows Compatible.....$24.95

1992 CoCoFest SIMULATOR by Allen C. Huffman

Graphics "adventure" based on the 1992 Atlanta CoCoFest

The next best thing to having been there! Digitized graphics of the event and a text command parser (ie, "get the box of disks") let you see all the vendors and even run into some famous faces of the CoCo Community. The show area, seminar room, and portions of the hotel are all represented. No true "goal", but you do have to figure some things out, like how to get into the show and how to buy items from vendors. Runs on a 640x192 hi-res graphics screen.

OS-9 Req: 512K CC3, OS-9 Lvl 2, 490K Disk Space...$9.95
OS/K Req: MM/1 or 100% K-Windows Compat.....$14.95

Worlds at War: ***A complete wargame simulator package!***

Finally, this Canadian masterpiece is available. Icon editor lets you build full color game pieces. Map editor lets you put together a multi-screen playfield. Options such as pass, move, attack, status, cargo, search, and build make this game a real "blast".

OS-9 Req: 512K CoCo3, OS-9 Level 2.....$SOON!

SUB-ETHA SOFTWARE

P.O. Box 152442,
Lufkin, TX 75915

Please include $2.50 S&H per order

Northern Xposure

'Quality Products from North of the Border'

OS-9 Level II

Smash! CoCo3 Arcade Game $29.95
Reviewed in 01 FEB 94 "68' micros"

Thexder:OS9 $29.95
Send manual or rompak to prove ownership

Matt Thompson's SCSI System $25.00
OS-9 Level II hard disk drivers, 256 & 512 byte sector support

Rusty Lauch ECB Progs from OS-9 $20.00

Disk Basic

Color Schematic Designer $35.00
Reviewed in 01AUG 93 "68' micros"

Oblique Triad Software
Various titles, order catalog for list & prices

OS-9/68000 (OSK)
Write for free catalogue

All prices in U.S. funds. Check or MO only. Prices include S&H

7 Greenboro Cres
Ottawa, ON K1T 1W6
CANADA
(613)736-0329

ADVERTISER'S INDEX:

For superior OS-9 performance, the

SYSTEM V

Provides a 68020 running at 25 MHz, up to 128 MBytes of 0 wait-state memory, SCSI and IDE interfaces, 4 serial and 2 parallel ports, 5 16-bit and 2 8-bit ISA slots and much more. The SYSTEM V builds on the design concepts proven in the SYSTEM IV, providing maximum flexibility with inexpensive expandability. Now available at 33 MHz.

AN OS-9 FIRST - the MICROPROCESSOR is mounted on a daughter board which plugs onto the motherboard. This will permit inexpensive upgrades in the future when even greater performance is required.

G-WINDOWS benchmark performance index is 0.15 seconds faster with a standard VGA board than a 68030 running at 30 MHz with ARTC video board (85.90 seconds vs 86.05 seconds).

Or, for less demanding requirements, the

SYSTEM IV

The perfect low cost, high-quality and high performance OS-9 computer serving customers world-wide. Designed for and accepted by industry. Ideal low-cost work-station, development platform, or just plain fun machine. Powerful, flexible, and inexpensively expandable. Uses a 68000 microprocessor running at 16 MHz.

Both computers provide flexible screen displays in the native mode with the optional VGA card.

Eight text modes are supported: 40 x 24, 80 x 25, 80 x 50, 100 x 40, 132 x 25, 132 x 28, 132 x 44, and 132 x 60. Foreground, background, and border colors are user selectable from up to 16 colors.

Eleven graphics modes are supported: 640 x 200 x 16, 320 x 200 x 256, 640 x 350 x 16, 640 x 350 x 256, 40 x 480 x 16, 640 x 400 x 256, 800 x 600 x 16, 640 x 480 x 256, 1024 x 768 x 16, 800 x 600 x 256, and 1024 x 768 x 256

Text and graphics modes may be selected by a utility provided, MODESET, by software using SetStt calls, or by termcap entries. In the text mode, the screen responds to standard VT100 control sequences. The full character set from Hex 20 through Hex FF is supported in the text modes up to and including 100 characters wide. The upper 128 characters follow the 'IBM Character Set 2' popular with many terminals and printers. These may be displayed on the screen by using the 'Alt' key and one or two other keys (software permitting).

Optional G-WINDOWS provides three screen resolutions: 640 x 480 x 256, 800 x 600 x 256, or 1024 x 768 x 256. You can have two full size 80 x 25 windows with room to spare. Or, a window as large as 122 x 44 using the large fonts or one over 180 x 70 using the small fonts

delmar co

PO Box 78 - 5238 Summit Bridge Road - Middletown, DE 19709
302-378-2555 FAX 302-378-2556